J Inf Process Syst, Vol.14, No.4, pp.1010~1032, August 2018
https://doi.org/10.3745/JIPS.04.0083

ISSN 1976-913X (Print)
ISSN 2092-805X (Electronic)

JOURNAL OF INFORMATION PROCESSING SYSTEMS JIPS

Restful Web Services Composition Using Semantic Ontology for Elderly Living Assistance Services

Sheik Mohammad Mostakim Fattah* and Ilyoung Chong*

Abstract

Recent advances in medical science have made people live longer, which has affected many aspects of life, such as caregiver burden, increasing cost of healthcare, increasing number of disabled and depressive disorder persons, and so on. Researchers are now focused on elderly living assistance services in smart home environments. In recent years, assisted living technologies have rapidly grown due to a faster growing aging society. Many smart devices are now interconnected within the home network environment and such a home setup supports collaborations between those devices based on the Internet of Things (IoT). One of the major challenges in providing elderly living assistance services is to consider each individual's requirements of different needs. In order to solve this, the virtualization of physical things, as well as the collaboration and composition of services provided by these physical things should be considered. In order to meet these challenges, Web of Objects (WoO) focuses on the implementation aspects of IoT to bring the assorted real world objects with the web applications. We proposed a semantic modelling technique for manual and semi-automated service composition. The aim of this work is to propose a framework to enable RESTful web services composition using semantic ontology for elderly living assistance services creation in WoO based smart home environment.

Keywords

Internet of Things, Service Composition, Web of Objects

1. Introduction

With the ongoing developments in medical science, people are now living longer than they did in previous generations. In fact, it is expected that around 20% of the world population will be age 60 or older by the year 2050 [1-3]. This may bring about many effects, including as increasing cost of healthcare, increasing disease, shortage of care givers, dependency, etc. In the United States, approximately 80% of the people who are over 60 are living with at least one chronic disease, and there are an estimated 5.4 million senior citizens suffering from Alzheimer's disease [4-6]. Almost 89% of older adults prefer to stay in the comfort of their own homes, so keeping this in mind along with attempting to reduce the costs they are providing for nursing, it is essential to develop technologies that help older adults age in place.

Depressive disorder like geriatric depression is also one of the major conditions often found in elderly

Manuscript received July 17, 2017; first revision September 19, 2017; accepted October 23, 2017.
Corresponding Author: Ilyoung Chong (iychong@hufs.ac.kr)
* Dept. of CICE, Hankuk University of Foreign Studies, Yongin, Korea (sm_fattah@yahoo.com, iychong@hufs.ac.kr)

persons [7,8]. A few of the major symptoms of depressive disorder are feeling sadness, feeling worthlessness, irritability, fatigue, crying spells, apathy, restlessness, lack of concentration, withdrawal, sleep problems, etc. One of the possible solutions for these phenomena is to introduce living assistance services in the smart home environment. In addition, the types of services required for different people can vary widely. For this reason, it is required to provide a semi-automated solution for the service creation mechanism of living assistance considering individual needs. In order to cope with these situations, research on living assistance technologies in the smart home has gained momentum in last few years and has largely focused on Internet of Things (IoT) [9-11]. The IoT combines a large number of technologies and comprises a variety of things or objects that are able to interact with each other and cooperate with their neighbors so as to reach common goals through unique addressing schemes and standard communication protocols [12-14]. The IoT platform promises to be a pervasive and ubiquitous networking platform to represent everyday objects such as pacemakers, sidewalks, traffic lights, and daily commodities as identifiable, readable, addressable, and controllable objects on the internet. Conventional smart home environments were mainly focused on approaches for accessing heterogeneous home devices. The recent changes and requirements are based on the virtualization of physical things, as well as collaboration and composition of the services provided by these physical things. Most of the IoT-based applications are built based on the services provided by sensors and actuators. An efficient and scalable approach is required to compose these services and to create collaboration models among these physical objects, which is one of the biggest challenges in IoT. In order to meet these challenges, Web of Objects (WoO) [15-17] focuses on the implementation of fetching the various real-world objects to the web application by enabling smart and intelligent services. The general goal of WoO is to simplify object and application deployment, maintenance and operation of IoT infrastructure through virtualization, and composition and collaboration of real world objects [18,19]. The WoO supports device-to-device collaboration, service compositions, and context-awareness [20]. It also supports sharing and the reuse of information, as well as using it outside of its domain where the information was created through the use of semantic ontology. In WoO, real-world entities are represented by virtual objects (VO) and a group of VOs engaging in accomplishing a special task is represented by composite virtual objects (CVO). A CVO works as the service entity in WoO which enables the creation of elderly living assistance services in smart home environments based on individual user requirements.

For elderly living assistance services in a smart home environment, it is required to provide support for disable or elderly persons in their daily lives so that they can do their work, which can be very simple sometimes but other times might require sequences of a few actions based on certain conditions. On the other hand, it is also required to continuously monitor those persons and to provide their family members or healthcare professionals with updated information in order to determine their physical and mental conditions. Based on the opinions of healthcare professionals, it may be possible to determine some health problems in advance, and based on their recommendation, it might be possible to overcome these problems or reduce the associated loss.

Considering the need for elderly living assistance services in smart homes, various solution have been proposed to ease the creation of such services. However, each person has different needs. Usually, living assistance services are application requirements specific. Sometimes, it is required to provide a mechanism for the further creation of service features based on user needs. The solutions that have been proposed so far do not consider the functional limitations associated with age. Motivated by these trends, in this work, a WoO based smart home network architecture is proposed so as to support elderly

living assistance services. The WoO based smart home architecture to support living assistance service enables RESTful web services composition with the help of semantic information. The composition technique is adapted from the classical artificial intelligence (AI) problem's solution Graphplan [21] algorithm. Throughout the work, static and dynamic CVO creation model along with their semantic representation are proposed. The contributions of this work are summarized as follows:

- An efficient and scalable CVO representation model for service composition is developed, which will ease the development of application through object collaboration and service composition.
- A semi-automated and dynamic as well as static and predefined CVO creation mechanisms are developed in order to build service features for smart homes.
- The proposed system architecture offers service providers with the ability to deploy new services and applications.

The rest of the paper is organized as follows: related works in this field are discussed in Section 2. Existing research efforts are presented for the creation of IoT-based infrastructures. In Section 3, semantic modelling and composition in WoO to support smart homes is described, which is required to come up with the solution of this work. The proposed system model and operation details of the different components are discussed in Section 4. Section 5 describes the experimental study, implementation specification, and analysis of this work. Finally, in Section 6, the summarization of this work and future scope are discussed as conclusion.

2. Existing Device Discovery Scheme

Providing elderly living assistance services in smart home networks has recently become a significant area of interest for researchers. It has also attracted the attention of many industries and academics. The major focus area of this work is to propose a solution based on IoT infrastructure to support living assistance services in smart home networks. So, the following section focuses on major research efforts to develop IoT platform or infrastructure to support heterogeneous applications.

IoT-A [22] is a European Lighthouse Integrated Project addressing the Internet of Things Architecture which proposes the creation of an architecture reference model together with the definition of an initial set of key building blocks. The IoT Reference Model gives the highest abstraction level for the definition of the IoT Architectural Reference Model. It encourages a common understanding of the IoT domain. The description of the IoT Reference Model consists of a general discourse on the IoT domain, an IoT Domain Model as a top-level description, an IoT Information Model explaining how IoT knowledge is modelled, and an IoT Communication Model in order to understand the specifics of communication between many heterogeneous IoT devices and the Internet. The definition of the IoT Reference Model conforms to that of the OASIS reference model definition. However, the project does not consider the semantic operability issues in the IoT environment.

The Smart Experiences (DiYSE) [23] project is an ITEA2 project which focuses its efforts toward enabling people to take control of their smarter surroundings, both at home and outdoors, ultimately evolving towards mass creativity in an open IoT world. This is to allow them to leverage aware services and smart objects to obtain highly personalized, social, interactive, and flowing experiences both at home and in the city.

The BUTLER [24], a 3-year-long European Union FP7 project has made a great contribution to the IoT research in Europe. Providing a user-oriented context-aware IoT platform has been the main focus of this project. With this motivation, this project has provided an integrated architecture for context-aware IoT application, cutting across communication layers, integrating location, security and behavior modelling, and addressing horizontal application domains. The project has acknowledged the transformative effects of IoT on many aspects of our societies from homes to hospitals, transport to shopping centers. In the BUTLER project, dynamic service composition is not considered while proposing the framework.

The term "iCore" [25] stands for internet connected objects for a reconfigurable eco-system, which is an EU FP7 project started on October 2011, and its duration was 36 months consisting of 8 countries with 20 cooperation industries. The iCore initiative mainly addresses two key issues in the context of the IoT, namely how to abstract the technological diversity that derives from the vast amounts of heterogeneous objects, while enhancing the reliability and determining how to consider the attitude of different users/stakeholders (owners of objects & communication means) for ensuring proper application provision, business integrity and, therefore, maximize exploitation opportunities. At the core of this level is the VO information model that describes the VOs. The VO information model has been previously mapped to the IoT Architecture Reference Model and SSN sensor ontology and it is categorized as a general semantic model for IoT actuators and sensors. The naming and addressing of VO is solved using approaches similar to the WWW, namely Universal Resource Identifiers (URIs). At the heart of the VO execution is the VO container, and it hosts VO front-ends (namely MQTT and HTTP REST) and various ICT device-specific back-ends, depending on the use cases and trials. Security interceptors and associated policies are in place with Policy Enforcement Point so as to prevent the ICT devices from being accessed by unauthorized entities (e.g., CVO). VO container also hosts various default and optional functionalities. In iCore, VOs and CVOs are implemented as software and their composition is mainly focused on policy-based creation.

There have recently been some other research efforts [26,27] towards designing IoT system architectures for different domains such as building energy management, smart grid, etc. However, there is no single standard which is aimed at implementing interoperable health service. Though technologies already integrated in everyday life, interoperability among health systems do not accomplish a proper solution.

In contrast from these approaches, this work proposes an architecture for the creation of living assistance services in WoO based smart homes which enables the automatic detection of the status of elderly person and enables user-defined service creation, overcoming the problems of the existing systems. To the best of our knowledge, no previous work has focused on providing a framework for the creation of elderly living assistance services in the smart home environment based on WoO utilizing the existing infrastructures. Thus, our proposed system provides elderly living assistance services as well as options for the creation of new services using CVO. In the next section, we will discuss the various technologies and methodologies that we have adopted in our system.

3. Semantic Modelling and Service Composition in WoO

Physical objects are virtualized and represented through the VOs, and VOs are composed together so as to provided application specific services in a smart home environment. In this section, VOs and

CVOs as well as their representations are going to be described. The composition of CVO is the most important part of this work. In providing living assistance services in a WoO based smart home environment, CVO plays the most important role.

3.1 Semantic Modelling of VO and CVO

Physical objects are defined and semantically represented using OWL instances in order to represent VOs. Initially, domain experts should analyze the system in order to create ontology for the purpose of creating instances of VOs. In order to enable composition, some specific representation of VOs and the functionalities they provide is required. Each of the real world entities is represented as VO in WoO. Fig. 1 depicts the semantic ontology model used to create VO and CVO OWL classes. In Fig. 1, not all of the properties are described. Few of the mandatory properties to enable composition are shown here. Each VO provides some functionalities. By using these functionalities, a VO can make changes to its own status or to the environment. This is why a VO state is included in the template to represent the current state of the VO. For example, a light VO can be turn off by its switch off function.

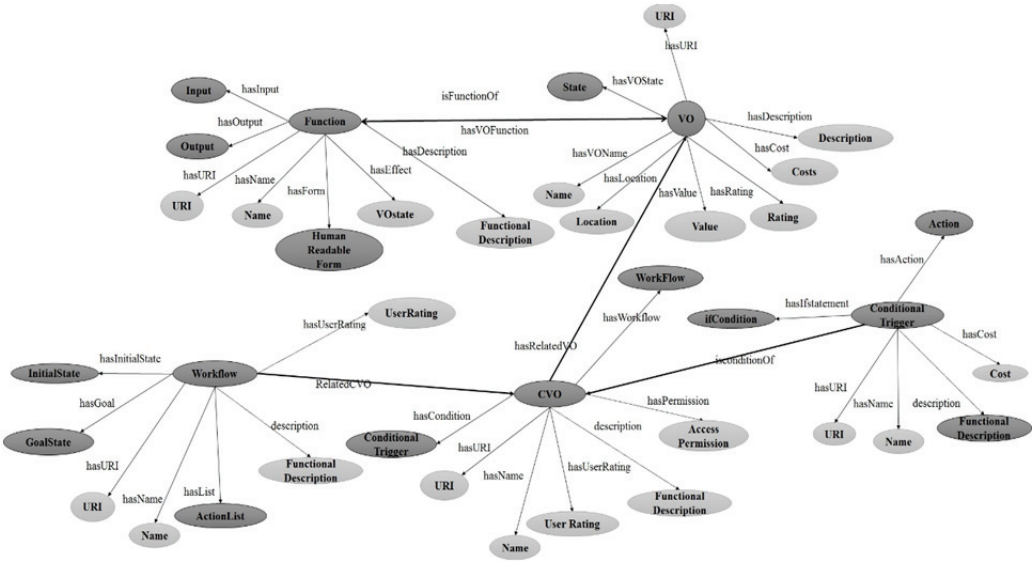

Fig. 1. Semantic modelling for VO and CVO representation.

Each of the VOs provides some functions such as `get_body_temp()`, `turn_on_light()`, etc. In order to use the functions or services provided by VOs in a composition, it is mandatory to represent their semantic information. A function is also required to be represented in both human-readable format and machine-readable format. When a service developer chooses some functions to be used in a composition he needs a human-readable format for feasibility.

Fig. 1 shows a VO function model as well. This figure also includes only a few of the important properties. Each function might have some inputs and outputs. A function also has a property called hasForm to represent it in human readable form and in STRIPS [28] like a format to be used in *graphplan* algorithm. Each function must have specific effects through which it changes a VO's state. Any property for the precondition of a function is not added intentionally to give the developer the

flexibility to create conditional services for living assistance in WoO-based smart homes. A function is nothing but an action which can have some consequences or effects on the environment. Sometimes, in order to execute a function, some preconditions must be met.

A CVO represents a group of objects assigned to perform a composite or complex task. In order to reach our goal, two types of CVOs have been proposed in this work. Based on the mechanism of creation, CVOs can be categorized as either (1) predefined and static CVO or (2) semi-automatic and dynamic CVO. Fig. 1 shows a generic ontology model of CVO to support both types of CVO in a WoO-based IoT environment. In the figure, only few mandatory properties of CVO are shown such as workflow property, conditional trigger property, related VO, URI, etc.

- *Predefined and Static*: A domain expert initially creates the template based on his/her analysis on the service requirements of the application domain. He or she might as well instantiate and deploy this type of CVO when the system is deployed, or it can be instantiated later by service request.
- *Semi-automatic and Dynamic*: One of the major goals of this work is to provide a mechanism where user specific service provision is required. In order to enable this mechanism, this kind of CVO is proposed. Based on the service request and using a composition algorithm, these CVOs are created. Based on the type of functionalities of CVO, we categorized CVO into either (1) workflow based or (2) conditional triggering based CVOs.
- *Workflow-Based CVO*: Workflow-based CVO is required when a service request needs a set of actions to be performed in a specific order so as to reach a specific goal. This CVO falls into the semi-automatic category. The workflow for this CVO is generated by the *graphplan* algorithm.
- *Conditional Triggering-Based CVO*: Conditional triggering-based CVO is applied when a service needs to monitor or wait for an event and then perform some tasks when that predefined condition is met. This CVO can be categorized as either predefined CVO or static CVO.

In order to create these two types of CVOs, it is required to develop the template for workflow and the conditional triggering template. When these two templates have been created, it is considered that CVO should be executed in a very simple manner. For this reason, some additional fields are added in these templates. A workflow is the order of execution of VO function in a CVO. A workflow is composed by the execution of the *graphplan* algorithm. A CVO workflow model also contains the initial state and a goal, which are required in the time of composition and execution. A conditional trigger model is created to execute a conditional action. A conditional triggering template contains the VO state and actions to be performed based on a certain condition. For example: *auto_temp_ctrl* template is a template that has related VOs such as temperature sensor, fan, and heater, and the condition can be: IF(sleep(elderly)) THEN turnoff(light).

3.2 Enabling Composition using *graphplan*

graphplan is an efficient algorithm [29-31] for automated planning that can also be mentioned as a simplified planning model developed by Avrim Blum and Merrick Furst in 1997. One of the most important things about *graphplan* is that it is a propositional planner, which means it has no variables. The input of the *graphplan* is a planning problem, from which it produces a sequence of operations for reaching a goal state, if possible. The planning problems are usually expressed in STRIPS. The number

of searches needed to find the solution from a straightforward exploration of the state space graph is reduced by using the *graphplan* algorithm, as it provides a novel planning graph. A planning graph is organized into layers. It consists of a sequence of levels that correspond to time steps in the plan. There are two kinds of layer alternatives in a planning graph:

- Literal: In a planning graph, hypothetical properties are represented using the literal. A level might contain a literal if it has been produced by an action of the previous level. There are persistence actions that handles the literals that remain true across all plans.
- Action: If the literals in an action's precondition are matched by the literals of a level and the action can be in that level, then it might be possible to perform the desired action

The start state that is the initial state is considered to be level 0. Each level might contain one or more actions. For a solution plan, at level *n* the literals will correspond to the goal state. The alternating layers of ground literals and actions are as follows:

- Nodes at action-level *i*: actions that might be possible to execute at time *i*.
- Nodes at state-level *i*: literals that might possibly be true at time *i*.
- Edges: preconditions and effects.

While forming the state and action level of the planning graph State-level 0 consist of all the atoms that are in the initial state of the planning graph and the negation of all of the atoms that are not in the initial state. Action level *i* consist of all the actions whose preconditions are satisfied in and non-mutex in state level *i*-1. Accordingly, state level *i* is comprised of all the effects of the actions in action level *i*-1. One of the important things here is the mutual exclusion in *graphplan* that is the calculation of whether two states are mutually exclusive (mutex) or not. In order to use the *graphplan* algorithm for composition, services provided by physical objects in the smart home are represented along with their semantic information in the form of VO function, and the physical objects are represented as VO. Each VO has some states which are mapped with the state of *graphplan* and VO functions are mapped with the actions of the *graphplan*.

4. Smart Home Architecture for Elderly Living Assistance

In order to enable elderly living assistance services in WoO-based smart home environments, it is required (1) to represent VOs with their functionalities to enable composition, (2) to semantically represent CVO for service creation, and (3) to use an efficient and scalable mechanism for CVO creation. We introduce a *graphplan* model to enable semi-automatic composition of workflow for CVO. In order to use *graphplan*, it needs to have a semantic representation of each component that may take part in the composition. In the next section we are going to introduce each of the components that have been developed for this work.

The architecture of a WoO-based smart home to support elderly living assistance service features is shown in Fig. 2. It consists of the following components: a gateway works as middleware between the WoO-enabled smart home server and physical objects. The WoO enabled smart home server works as a platform for these works which provide support for the creation, deployment, and execution of VO and CVO. WoO-enabled home server also provides interfaces to the application server and CVO composition unit.

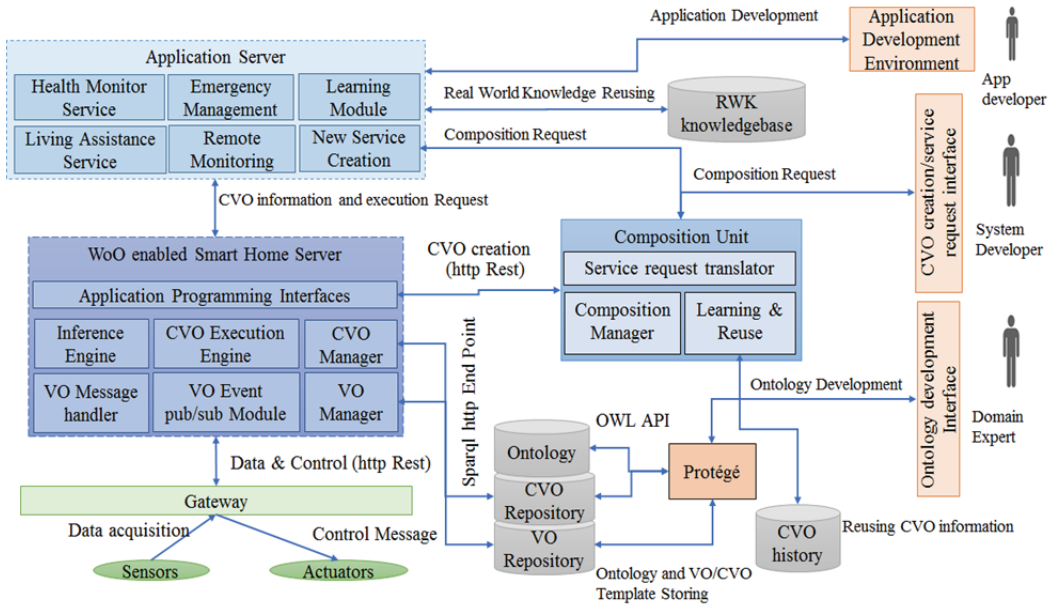

Fig. 2. Proposed system architecture.

CVO composition unit enables the creation of CVO based on user requests of service creation. The application server is built on top of the WoO-enabled smart home server. The application server provides application-specific services in the smart home, which in our case are elderly living assistance services. In this architecture we also include an ontology development tool in order to create an ontology for the smart home environment. There are also repositories for VO, CVO, and ontology. The composition unit has a CVO history database which is also used to store user requests for the creation of CVO. The application server has also a knowledge database to provide intelligent services.

The first step towards providing elderly living assistance service creation in WoO-based smart homes is to virtualize physical objects and represent them semantically along with their functionalities. The virtualization and semantic representation can be decomposed in the following sub-problems:

- Design ontology, VO, and CVO using an ontology development tool and deploy them in the repository,
- Collect data from the gateway and update VO, CVO,
- Perform reasoning on VO and CVO.

The next step is to enable composition to the service developer. This step can be divided into the following categories:

- Provide user with the CVO creation interface with available VO, CVO along with their functionalities and suggestions,
- Get CVO composition request, translate the request, and match with history,
- Compose the workflow for CVO and get user feedback on composition.

Finally, the applications are responsible for execution request of a set of CVO in order to provide elderly living assistance services. In the following section, we describe how elderly living assistance services can be composed in a WoO-based smart home environment.

4.1 Virtualization and Semantic Representation

In order to virtualize a physical object, a domain expert should analyze its properties and functionalities. Then it is required to design their ontology and instantiate the VOs. A predefined CVO to provide some common functionalities can also be created in this time. SWRL is used to define the rules for such a CVO. In the following section we are going to describe this process step by step.

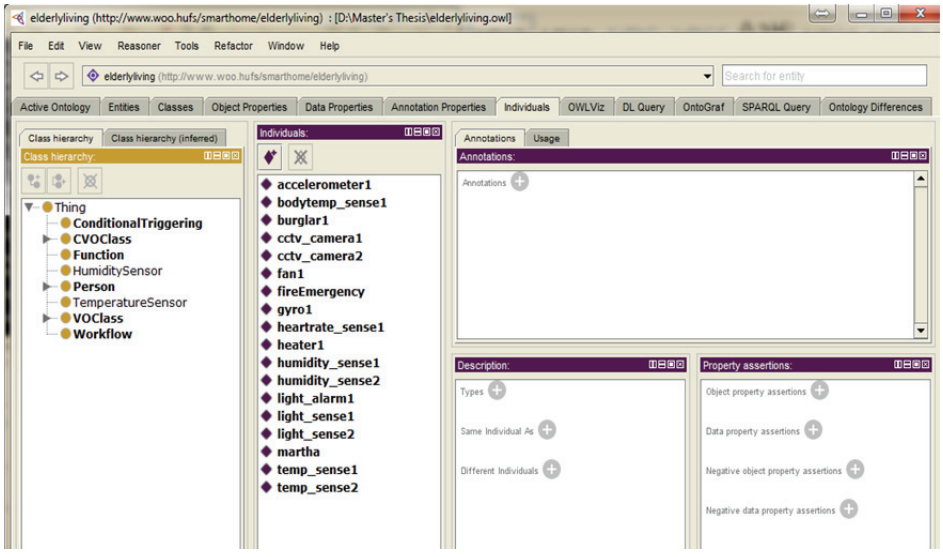

Fig. 3. Ontology development interface.

4.1.1. Ontology development

Fig. 3 shows the ontology development environment for this work. The ontology development tool (Protégé) provides a GUI for a user to create ontology.

In Protégé a user can create classes, properties, and relations between the classes. It is also possible to define SWRL rules and reasoning in protégé. Users can create ontology of RWO, VO, and predefined CVO. Everything is saved in the OWL format in a triple database. The triple database is connected to a SPARQL endpoint with the WoO-enabled smart home server. SWRL rules are also created through protégé, which is used by an inference engine on the request of the application server. Protégé also provides class expression development and SPARQL query and reasoner which are used for test development environment in this work.

4.1.2 Data collection and update VO, CVO

The gateway works as a virtualization middleware for our system architecture. All of the sensors and actuators are connected through the gateway to the WoO-enabled smart home. A gateway consists of all of the drivers that are required to receive and deliver data and control message to the sensors and actuators. A gateway is configured to add the URI of each sensor and actuator while receiving data from the physical objects. The gateway should also be capable of handling REST communication, or in other words. It should be capable of working as a web server.

4.1.3 Perform reasoning

After deploying the initial environment, a reasoning is performed in order to update all the status of VO and CVO based on the current environment. For example, during design, a VO: temp_sense1 is classified under temperature sensor. A temperature sensor is a subclass of VO. So, after reasoning, temp_sense1 become a subclass of VO.

4.2 Enabling Composition for User

In order to enable composition services to the user, a user is provided with a CVO composition interface with available VO and CVO information. The composition request is then passed to the composition unit. The composition unit either reuses existing CVO or compose the CVO. In the following section a brief discussion is provided.

4.2.1 CVO composition request

In order to provide semi-automatic CVO composition service to the user, it is required to provide a CVO composition request interface where the user can see the available VO and CVO, and the functions provided by them. A CVO composition unit is a web server which provides the user interface in HTML5 and JavaScript. Composition unit pulls information from the home server regarding VOs and CVOs and puts it in the composition interface. Users can request to perform two kinds of CVO compositions. In the case of predefined CVO, users should set the condition and the action of an existing CVO. On the other hand, users should set an initial state, a goal, and a set of action to compose a workflow based CVO. Fig. 4 shows a CVO creation request example.

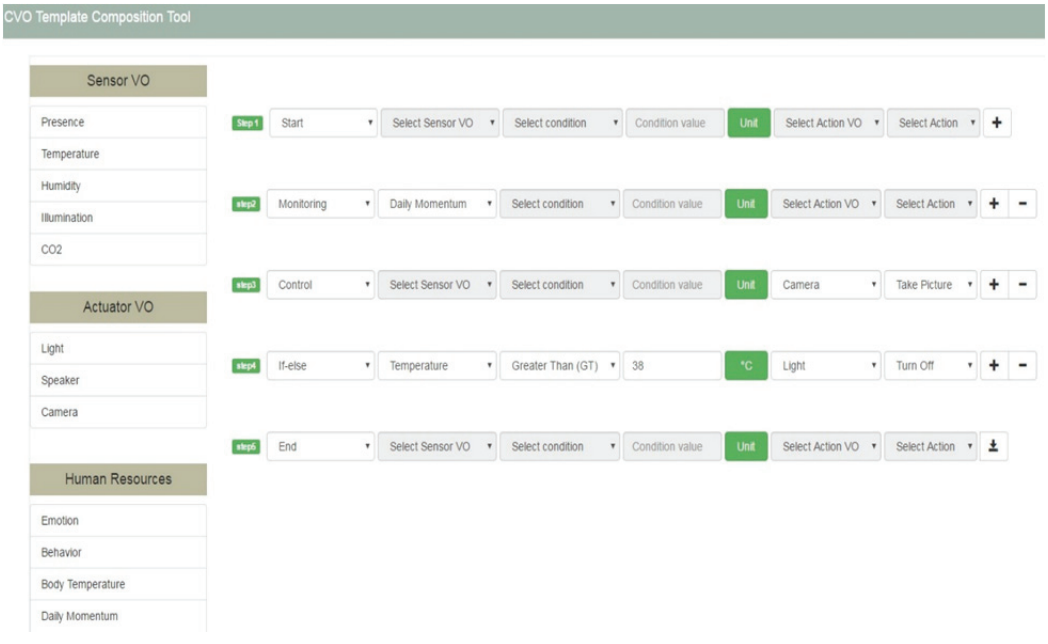

Fig. 4. CVO composition tool.

4.2.2 Request translation and reuse

Composition unit has a service translate module in order to interchange requests between human-readable format requests and machine-readable format using the semantic information of VO and CVO. These request are then passed to the learning and reuse module. The request is matched with the previous history and existing CVO information. If no suitable CVO is found to serve the request, it is passed to the composition unit.

4.2.3 Workflow based composition

The CVO composition unit is shown in Fig. 5. It consists of three modules, which are a service request interface, a service translator, and a rule-based learning and reuse module. The service request interface provides a user-friendly environment to VO and CVO and actions to create workflow-based CVO and set up parameters for conditional triggering-based CVO. A service request is sent to the composition unit in human-readable form. The service translator translates the service request in to a machine-readable format. The request is then passed to the learning and reuse module. The learning and reuse module first looks in the history of CVO database to search service requests that match the current request. If a reusing opportunity is possible then the existing CVO is provided to the user.

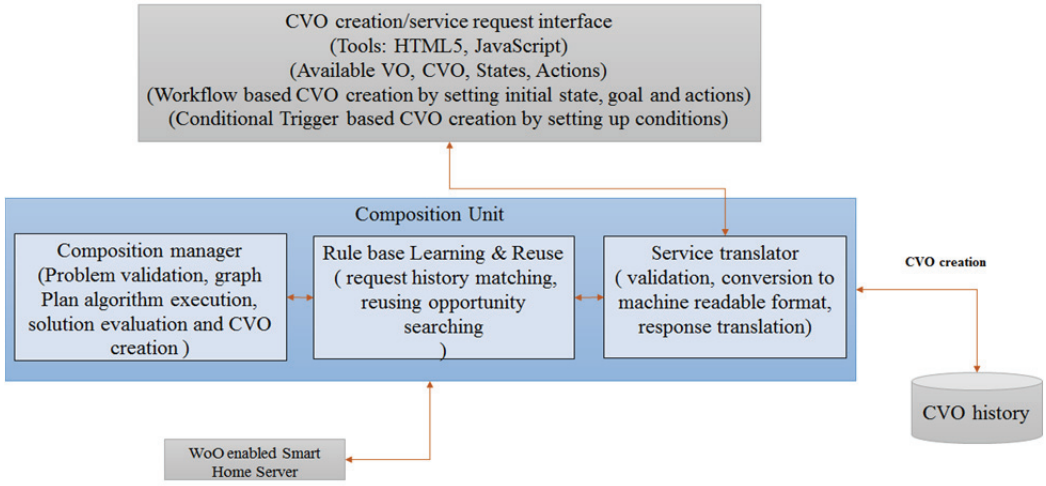

Fig. 5. Composition environment of CVO.

Each request is saved into the history database along with the composed CVO and user feedback. Each VO function rating is stored based on their frequency of use. When a user chooses a function for a specific service composition the functions with higher ratings are suggested to the user. If no history is found regarding the requested service, the request is passed to the composition manager where the composition of workflow for the CVO is composed using the *graphplan* algorithm. Afterwards the composition of workflow it is attached to the instantiated CVO. Each time a CVO is created, user feedback is taken so that it can be later used to suggest to the user for later requests. After the composition of a CVO is completed, it is delivered to the WoO-enabled smart home server in order to be stored and used later by the applications.

A sleep problem CVO which monitors whether an elderly person is having some problem in sleep or not. This CVO can be used in depression detection services as well. The following example shows sleep problem CVO workflow generation process. Table 1 illustrates the actions for workflow. Let Initial State: S0 = {Midnight} and Goal: G= {SleepProblem}

Table 1. Actions and their preconditions and effects

Action	Precondition	Effect
CheckPluse()	None	~lowPulse
noAccelerate()	None	LowAcceleration
Detect()	Midnight	~light
checkInso()	lowAcceleration, ~lowPluse	~sleep
Determine()	~sleep, ~light, midnight	SleepProblem

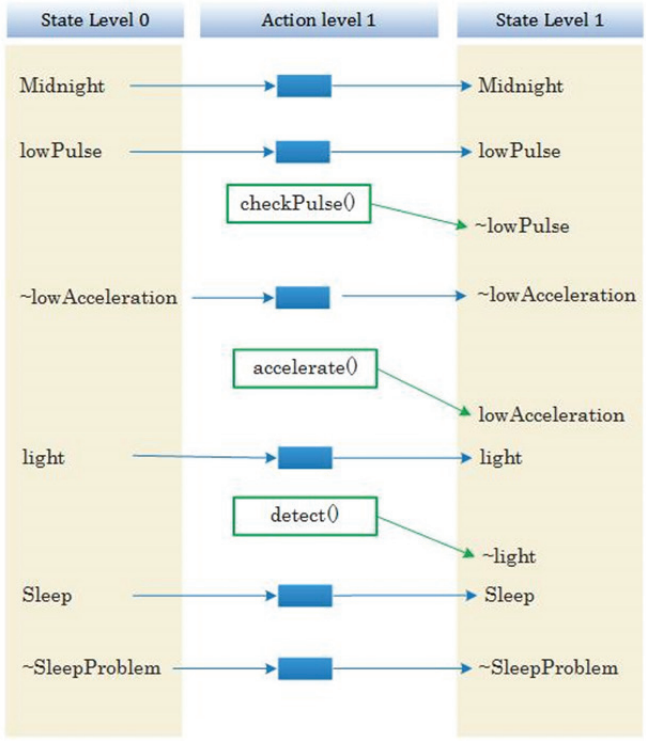

Fig. 6. Step 1: Sleep problem detection workflow.

Now, we have defined the composition problem in a classical AI planning or *graphplan* problem. The composition of the sleepProblem detection Workflow is described step-by-step in Figs. 6–8. The initial states S0 contains the initial goal and the negation of all the states that are not in the initial states.

Application services are enabled with the help of a WoO-based smart home server. The home server provides the application programming interfaces to the application server. The application server can request to use existing CVO or to create new CVO in order to provide intelligent service features to the user.

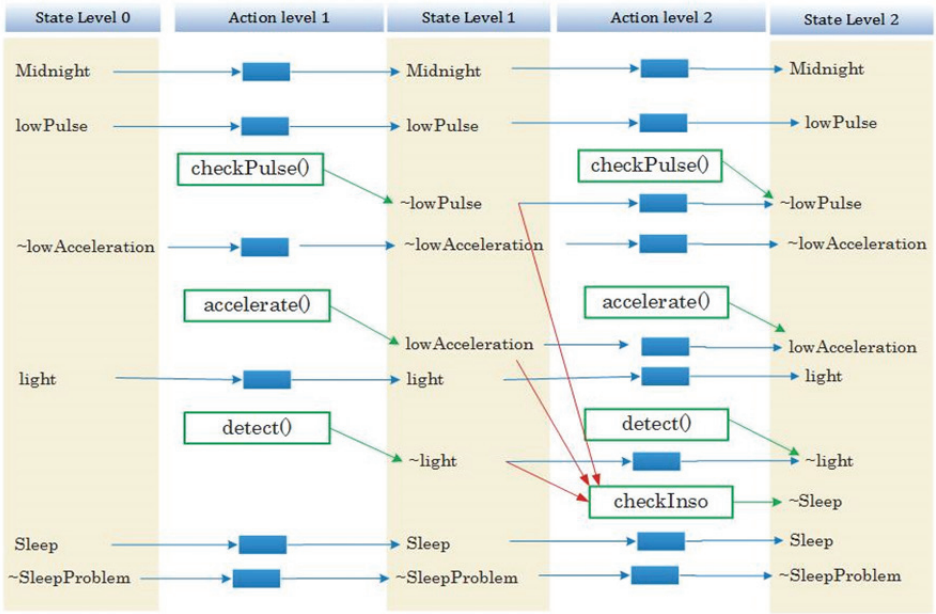

Fig. 7. Step 2: Sleep problem detection workflow.

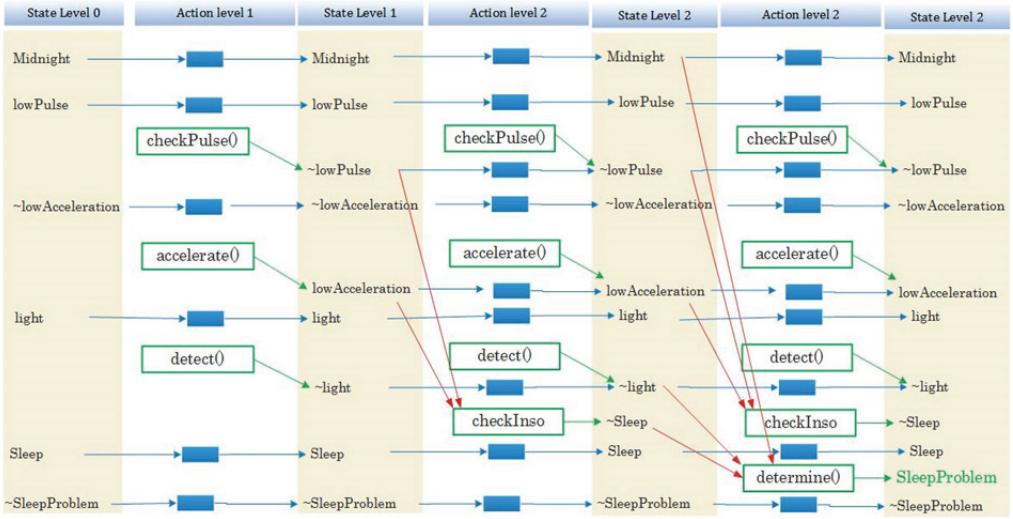

Fig. 8. Step 3: Sleep problem detection workflow.

4.3 Enabling WoO based Smart Home Service

This is where service using opportunity is provided to the applications. Although several WoO-based work has already introduced ontology servers as core platforms of the WoO architecture, we have adopted the idea of ontology server and improvised it to be used in this work. Fig. 9 depicts the architecture along with the responsibilities of each components. As we can see, there are six components distributed in VO and CVO level. The bottom part of the home server is connected to the gateway via REST communication. The home server receives and sends data as well as controls

instructions from or to the gateway.

There are three components in the regarding VO. The VO message handler basically handles all of the communication between the physical objects and the virtual objects. We also consider a pub/sub module of VO for event monitoring from the CVO level. The VO pub/sub module publish data to the specific subscriber and this mechanism enables the event monitoring in the WoO-based smart home environment so as to provide elderly living assistance services.

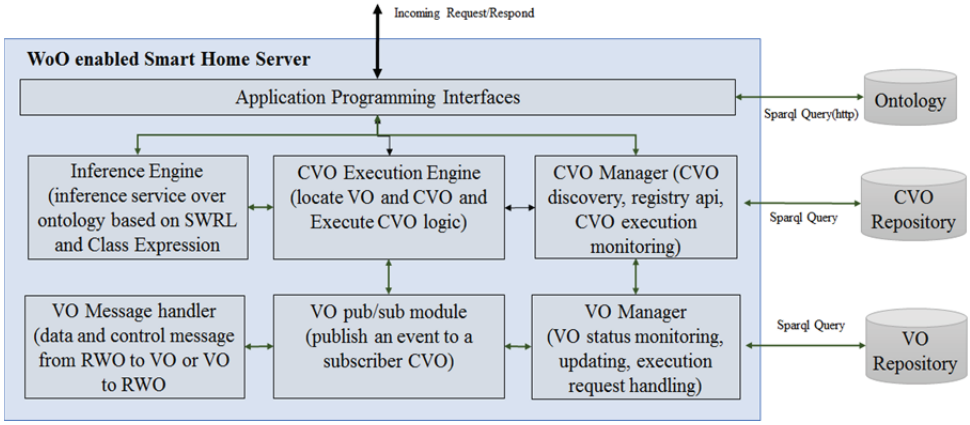

Fig. 9. WoO-enabled smart home server functionalities.

The VO Manager gives different types of services including request handling, VO status monitoring, and updating. It also handles execution requests between multiple requests. The VO manager provides restful web services in order to provide these services to the CVO and application layer.

The inference engine provides reasoning services to the application layer. A semantic reasoner is provided here. SWRL and Class expression are used to create rules during the system development or they can be created later from the application layer using SPARQL construct query. CVO execution engine executes a CVO based on the workflow logic or conditional triggering. During CVO execution, it also communicates to the VO manager in order to locate and execute VO functions. On the other hand, a CVO manager provides services like the discovery, registration, and handling of the execution of CVO.

The application server provides the application-specific service based on the WoO-enabled smart home server. In this case, the application server is responsible for providing elderly living assistance services. Rather than providing a few specific application services, our work mainly focuses on the creation of a mechanism of diverse application. In this work, instead of developing the application specific services we mainly created a scope for the creation of application using CVO. Our envisioned architecture for application server is shown in Fig. 10. The composition unit of the system architecture provides application programming interfaces to the application server in order to create CVO using service requests. An application server can create new service as well use the existing services through the creation of a service graph using CVOs. The application server can communicate with the home server and the CVO composition unit; it learns from the real world to create knowledge based on monitoring CVO execution, and stores this knowledge in a knowledge base in order to provide intelligent application-specific services using AI mechanisms.

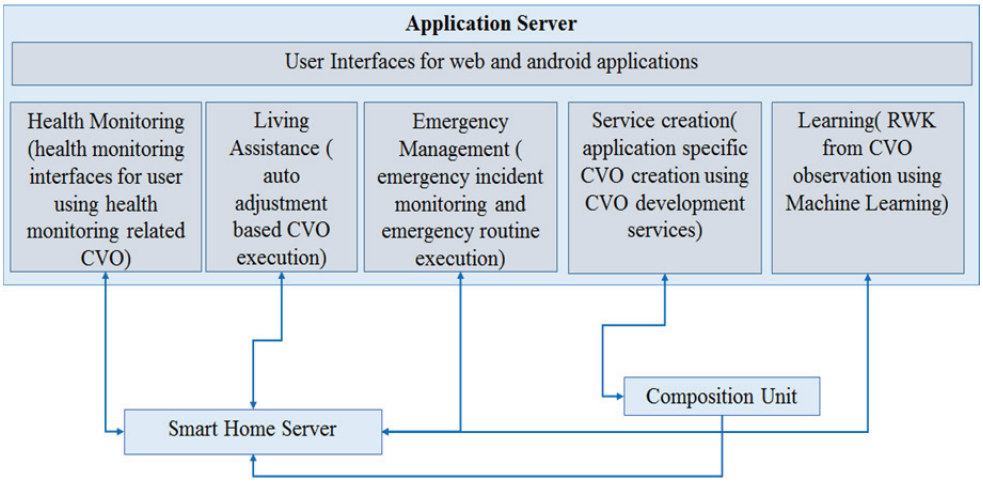

Fig. 10. Application server functionalities.

For example, if an application service is for depression detection in an elderly person: In order to detect depression for elderly person there are several symptoms that an application has to monitor (Fig. 11). Let us consider three CVOs which monitor the symptoms for depression. CVOs for Depression detection services include Sleep Problem Detection CVO, Physical Pain Detection CVO, and Apathy Detection CVO.

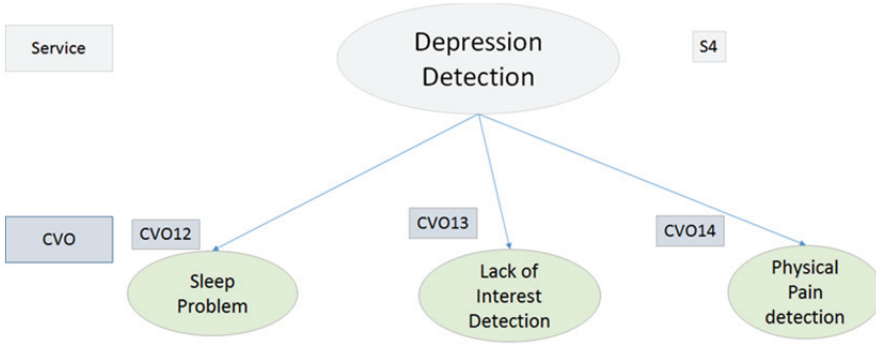

Fig. 11. Service creation with CVO.

By using these CVOs, an application can decide whether an elderly person is depressed or not. In this section, we have presented the system architecture in detail in order to provide elderly living assistance in a WoO-based smart home environment. This work provides a complete system framework to support elderly living assistance services as well as the creation of these services in a semi-automatic and manual approach. The natural language processing service can be easily integrated into the system framework, which in turn can provide the user with a more flexible way to create service requests to the user. Moreover, we can also provide android based services to the service consumers for monitoring and performing action from the applications. Therefore, our proposed system framework can be extended to include further services, making it a scalable solution. Further, the methodologies and technologies that have been conducted in our system makes this a light-weight and flexible solution.

5. Prototype Implementation

In this section, we are going to briefly describe the prototype implementation scenario and analyze the system in order to prove the effectuality of the proposed system. We begin with a description of the scope of prototype implementation including the tools that have been used for the composition of CVOs. We then study the result of our prototype implementation.

5.1 Prototype Design Scope and Consideration

In this work, our major focus is on the development of CVO in order to provide service creation of elderly living assistance for a WoO-based smart home environment. This why, rather than designing the complete system, we focused on the most significant parts of the proposed system in order to verify our ideas, which are:

- Ontology development using protégé,
- VO and predefined CVO development in protégé,
- Workflow development using *graphplan* algorithm,
- Elderly living assistance CVOs for living assistance.

5.2 Environment Setup

We are going to evaluate our proposed system by conducting a pseudo-experiment. We consider a smart home environment where there exist few sensors and actuators, as well as an elderly person, according to our assumption. Tools that have been used for the development are described as follows and Table 2 shows the list of sensors and actuators for this work.

- Gateway (web server): In our experiment, the gateway is implemented as a web server which serves data to the home server, and the home server also can send requests to the gateway. In order to provide data to the Smart Home Server, it provides RESTful API. For subscription to the sensor data it provides pub/sub mechanism using web socket. However, in our experiment, instead of using real time data, we have used simulated sensor actuator data in the gateway. According to WoO, the gateway should support REST, pub/sub, CoAP, DTLS, and other protocols.
- Ontology Development (protégé): Ontology development and VOs, as well as predefined CVO is created on the protégé. After the creation of these, everything is stored in the TDB (triple database) provided by the Apache Jena library using Jena Fuseki, which is a SPARQL endpoint.
- Smart Home Server (Java, Servlets, Apache Jena): Smart home server is developed in Java where application programming interfaces are developed by Java servlets. Apache Jena library is used for enabling semantic web technologies.
- Inference Engine (Pellet reasoner): Pellet reasoner library for Java is installed in the smart home server in order to provide the application with reasoning services.
- Composition Unit (Node.js, HTML5, JavaScript): The composition unit for CVO creation is built on Node.js server which is a very light-weight and event queue-based web server. HTML5 and JavaScript are used for the web interface for the user to request service creation or CVO composition.

- Databases (TDB, MongoDB): For storing the history of service request and CVO, we have used MongoDB, which is a NoSQL database. For storing VO, CVO, and ontology, we have used the triple database provided by Apache Jane libraries.
- Communication (HTTP REST): As our implementation supports all the communication and application programing interfaces in REST, developers can choose any kind of platform for the development they want. In addition, the RWK knowledge database is optional for the developer which will be used for creation of knowledge based on learning and for creation of intelligent services.

Table 2. The list of sensors and actuators considered for user, room inside and outside

User sensors	Room inside sensor and actuators	Outside sensors and actuators
Accelerometer sensor	Light sensor 1	Light sensor 2
Gyroscope sensor	Temperature sensor 1	Temperature sensor 2
Body temperature sensor	Midnight	Humidity sensor 2
Heart rate monitoring sensor	Humidity sensor 1	CCTV camera 2
	CCTV camera 1	Led lamp 2
	Led lamp 1	Burglar alarm
	Heater	Buzzer
	Fan	Light alarm

5.3 Development Scenario

The first step of our development is to build ontology in protégé. Fig. 12 shows the developed ontology for the elderly living assistance in a smart home environment. All of the components of the smart home that are considered for this implementation are described in the ontology. The next step is to define VOs along with their properties. This is also done in protégé initially.

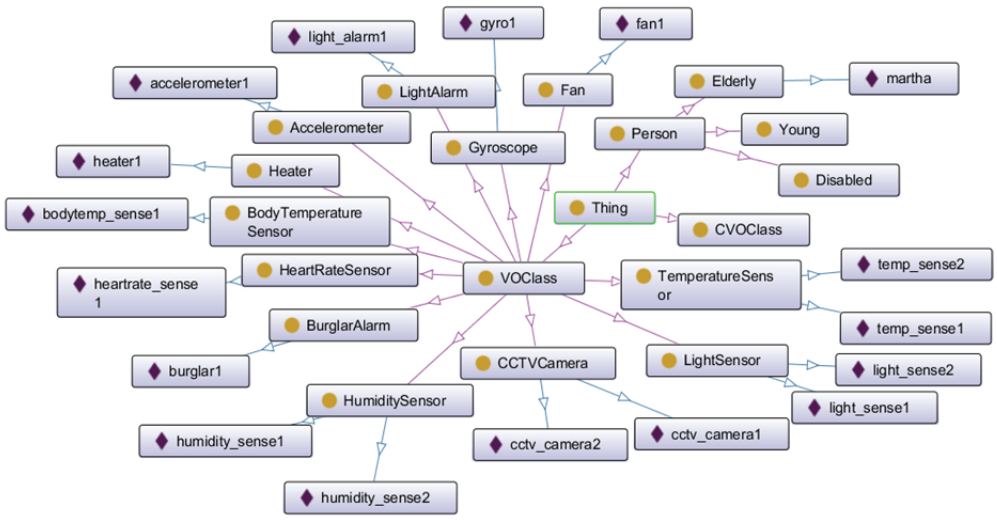

Fig. 12. Ontology of smart home.

After that, we are ready to define predefined CVO for elderly living assistance services. Fig. 13 shows an example of emergency CVO for fire detection which is build using SWRL rule. The ontology, VOs, and CVOs are designed in protégé then deployed in the TDB using Jena Fuseki.

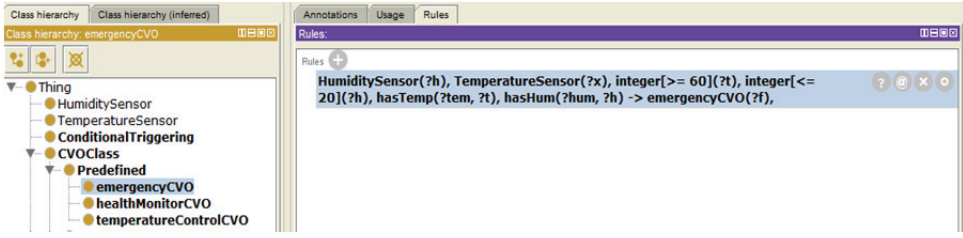

Fig. 13. Emergency CVO.

We then develop the workflow-based CVO using the *graphplan* algorithm. In order to compose the workflow we need to define an initial state, a goal, and a set of actions. We developed only the workflow for the CVO using a java library for *graphplan*. Our targeted problem is: turn of the light if Martha is sleeping. In order to create the workflow we have defined the problem as follows.

- Init(on(light))
- Goal (off(light))
- Action (checkAcceleration(Maria);
  - pre: none()
  - Eff: NoAcceleration(Maria)
- Action (checkPulses(Maria));
  - Pre: none()
  - Eff: HighPulses(Elderly)
- Action (TurnOff(light));
  - Pre: on(light) & sleep(maria)
  - Eff: off(light)

The generated workflow is:

- checkAC (maria)
- checkPulse (maria)
- checkSleep (maria)
- TurnOff (l1, maria)

Fig. 14 shows the *graphplan* simulation output for this workflow generation. Fig. 15 shows the intermediate simulation scenario for this workflow.

```
INFO: Running planner, maximum memory: 3,628.5MB
INFO: Expanding graph
INFO: Goals not possible with 1 steps
INFO: Expanding graph
INFO: Goals not possible with 2 steps
INFO: Expanding graph
INFO: Extracting solution
INFO: Plan found with 3 steps
INFO: Planning took 57ms ( 0s )
INFO: Total memory used: 245.5MB
INFO: Plan found:
```

Fig. 14. *graphplan* execution.

```
1
2  Proposition Layer
3  Proposition: on(l1)
4   Mutex with:
5   Action Layer
6  noopon(l1)
7   Mutex With:
8  checkPulse ( maria )
9   Mutex With:
10 checkAC ( maria )
11  Mutex With:
12 Proposition Layer
13 Proposition: on(l1)
14  Mutex with:
15 Proposition: noAC(maria)
16  Mutex with:
17 Proposition: lowPulse(maria)
```

```
16  Mutex with:
17 Proposition: lowPulse(
18  Mutex with:
19  Action Layer
20 noopon(l1)
21  Mutex With:
22 noopnoAC(maria)
23  Mutex With:
24 checkPulse ( maria )
25  Mutex With:
26 checkSleep ( maria )
27  Mutex With:
28 checkAC ( maria )
29  Mutex With:
30 nooplowPulse(maria)
31  Mutex With:
```

```
32 Proposition Layer
33 Proposition: on(l1)
34  Mutex with:
35 Proposition: noAC(maria)
36  Mutex with:
37 Proposition: sleep(maria)
38  Mutex with:
39 Proposition: lowPulse(maria)
40  Mutex with:
41  Action Layer
42 noopon(l1)
43  Mutex With:
44 noopnoAC(maria)
45  Mutex With:
46 checkPulse ( maria )
47  Mutex With:
48 checkSleep ( maria )
49  Mutex With:
50 checkAC ( maria )
```

Fig. 15. Intermediate results of execution.

Fig. 16 shows the CVO composition MSC diagram. Service requests were generated from the Composition unit interface. The request is translated into machine-readable format and learning and reuse module finds existing CVOs to match with the service request. If no match is found, the request is passed to the composition manager. Composition manager composes the workflow for the CVO. After a successful composition the CVO is passed to the CVO manager to the smart home server. Composition manager stores the CVO.

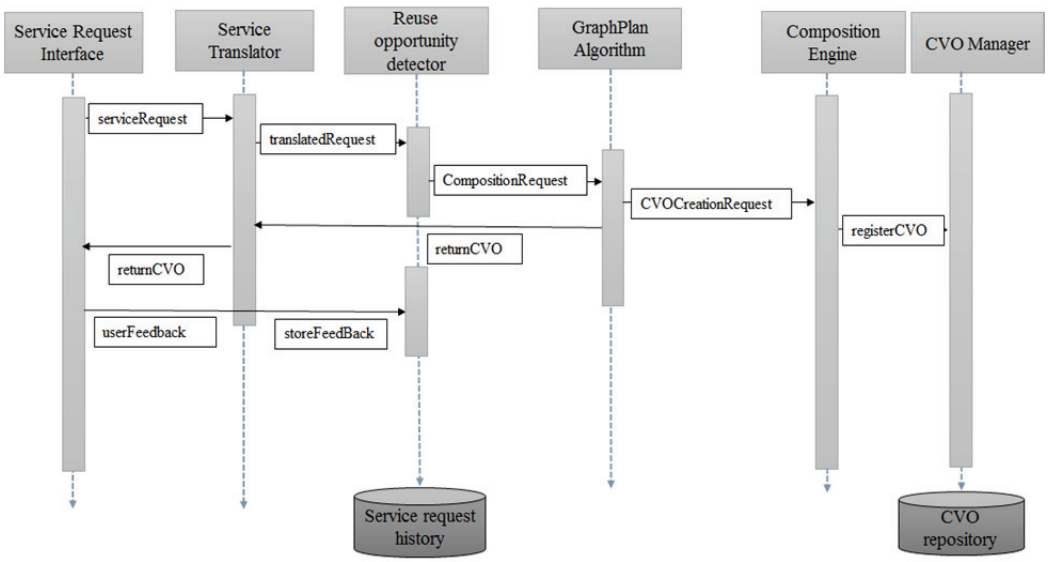

Fig. 16. Dynamic CVO composition MSC diagram.

Fig. 17 shows the execution request from the application server to the smart home server. The request is handled by the CVO manager and the execution is performed by the CVO execution engine. CVO execution engine executes the CVO logic. In order to execute the logic, the CVO execution engine locate the VOs from the VO manager and passes the request to the VO manager.

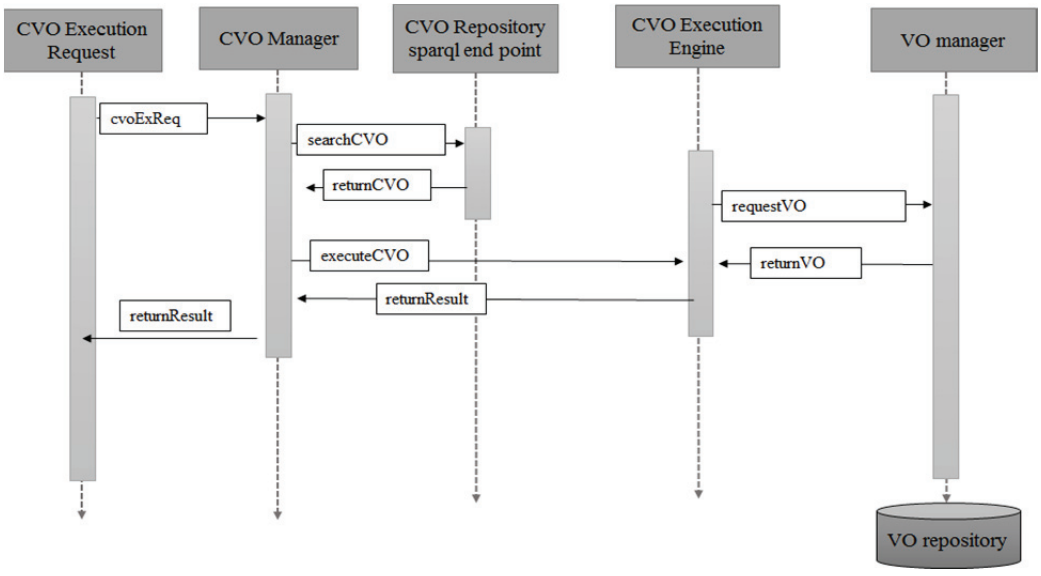

Fig. 17. MSC for CVO execution request.

5.4 Discussion and Analysis

As we have discussed earlier, instead of development of the complete system, we have only developed some significant parts of the total system to analyze our idea. We have converted CVO composition problem to a general workflow problem in order to solve it with a *graphplan* algorithm to create semi-automatic composition of CVO for elderly living assistance. With the combination of two kinds of CVOs it is possible to efficiently deploy elderly living assistance services. Workflow composition is a costly task this is why we have proposed to reuse the existing CVO instead of composing new CVO every time which will give a better performance. If we only use predefined CVO sometimes the system will fail to provide any CVO for a service request. On the other hand, if we only use the semi-automatic approach the cost would be high in term of composition time, memory, and response time of the service request. In this section we have discussed the implementation of our proposed idea and the feasibility of our approach. Instead of using only static or semi-automatic approach we have introduced a hybrid mechanism to create CVO for elderly living assistance services in WoO based smart home environment. In our future work, we are going to develop the simulation environment while introducing more CVO for elderly living assistance and composition tool to perform following analysis:

- Semi-automated Composition time and memory requirement,
- Failure rate comparison between static and dynamic CVO creation,
- Response time comparison between static and dynamic CVO.

6. Conclusions

We have designed a light-weight a scalable CVO representation and restful web services composition mechanism to support elderly living assistance services in WoO-based smart home environment. A

system architecture is proposed in this work as well. This work focused on creation of elderly living assistance services rather than the creation of one specific application service. It takes advantage of semantic web technologies and *graphplan* algorithm in order to represent VOs and CVOs as well as the composition of CVO from VOs. In this work, we have mainly focused on the CVO layer of our architecture as many previous works already examined how to collect data from physical objects and virtualization of physical objects. We have referred to the virtualization process from previous WoO-based research. Our proposed system and implementation analysis shows the feasibility of using a WoO-based approach in order to provide elderly living assistance services in a smart home environment and how CVO can cooperate in creation of service features. Our proposed system is mainly focused on the CVO representation and composition part. And our implementation also focused to the CVO representation and workflow composition part. The semantic ontology in our work is extendable. As there is not yet a published standard in how to incorporate ICT area, we did not focus on a specific data model for composition. It is also mandatory to keep the model as simple as possible in order to use it in semi-automatic composition. However, as soon as medical organizations publish their standards data model, it will be possible to translate the data in our proposed model by using an ontology map and translator. In the future we would like to provide a comprehensive analysis with the performance and feasibility of our proposed system. We would also like to provide some performance-related issues in the execution of the proposed CVO.

Acknowledgement

This work was supported by Hankuk University of Foreign Studies Research Fund of 2018.

References

[1] P. Rashidi and A. Mihailidis, "A survey on ambient-assisted living tools for older adults," *IEEE Journal of Biomedical and Health Informatics*, vol. 17, no. 3, pp. 579-590, 2013.

[2] J. Andreu-Perez, C. C. Poon, R. D. Merrifield, S. T. Wong, and G. Z. Yang, "Big data for health," *IEEE Journal of Biomedical and Health Informatics*, vol. 19, no. 4, pp. 1193-1208, 2015.

[3] M. Memon, S. R. Wagner, C. F. Pedersen, F. H. A. Beevi, and F. O. Hansen, "Ambient assisted living healthcare frameworks, platforms, standards, and quality attributes," *Sensors*, vol. 14, no. 3, pp. 4312-4341, 2014.

[4] U. Esnaola and T. Smithers, "Whistling to machines," in *Ambient Intelligence in Everyday Life*. Heidelberg: Springer, 2006, pp. 198-226.

[5] D. J. Cook, J. C. Augusto, and V. R. Jakkula, "Ambient intelligence: technologies, applications, and opportunities," *Pervasive and Mobile Computing*, vol. 5, no. 4, pp. 277-298, 2009.

[6] D. J. Cook and S. K. Das, "Pervasive computing at scale: transforming the state of the art," *Pervasive and Mobile Computing*, vol. 8, no. 1, pp. 22-35, 2012.

[7] B. S. Wiese, "Geriatric depression: the use of antidepressants in the elderly," *British Columbia Medical Journal*, vol. 53, no. 7, pp. 341-347, 2011.

[8] A. M. Koenig and M. A. Butters, "Cognition in late-life depression: treatment considerations," *Current Treatment Options in Psychiatry*, vol. 1, no. 1, pp. 1-14, 2014.

[9] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): a vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645-1660, 2013.

[10] G. Kortuem, F. Kawsar, V. Sundramoorthy, and D. Fitton, "Smart objects as building blocks for the Internet of Things," *IEEE Internet Computing*, vol. 14, no. 1, pp. 44-51, 2010.

[11] L. Da Xu, W. He, and S. Li, "Internet of Things in industries: a survey," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 4, pp. 2233-2243, 2014.

[12] L. Atzori, A. Iera, G. Morabito, and M. Nitti, "The social Internet of Things (sIoT)–when social networks meet the Internet of Things: concept, architecture and network characterization," *Computer Networks*, vol. 56, no. 16, pp. 3594-3608, 2012.

[13] A. Botta, W. De Donato, V. Persico, and A. Pescape, "Integration of cloud computing and Internet of Things: a survey," *Future Generation Computer Systems*, vol. 56, pp. 684-700, 2016.

[14] J. Jin, J. Gubbi, S. Marusic, and M. Palaniswami, "An information framework for creating a smart city through Internet of Things," *IEEE Internet of Things journal*, vol. 1, no. 2, pp. 112-121, 2014.

[15] M. G. Kibria, S. M. M. Fattah, K. Jeong, I. Chong, and Y. K. Jeong, "A user-centric knowledge creation model in a web of object-enabled Internet of Things environment," *Sensors*, vol. 15, no. 9, pp. 24054-24086, 2015.

[16] Z. U. Shamszaman, S. S. Ara, I. Chong, and Y. K. Jeong, "Web-of-Objects (WoO)-based context aware emergency fire management systems for the Internet of Things," *Sensors*, vol. 14, no. 2, pp. 2944-2966, 2014.

[17] S. M. M. Fattah, M. G. Kibria, K. Jeong, and I. Chong, "Knowledge driven architectural model to support smart emergency service in web of objects based IoT environment," *The Journal of Korean Institute of Communications and Information Sciences*, vol. 40, no. 2, pp. 408-418, 2015.

[18] ITEA3 Projects, "Web of Objects: D2.1 State-of-the-Art," [Online]. Available: https://itea3.org/project/web-of-objects.html.

[19] M. G. Kibria, K. Jeong, S. M. Fattah, and I. Chong, "Smart emergency service in WoO based shopping mall," in *Proceedings of 2014 International Conference on Information and Communication Technology Convergence (ICTC),* Busan, Korea, 2014, pp. 1001-1002.

[20] ITEA3 Projects, "Web of Objects: D3.3 WoO Reference Architecture," [Online]. Available: https://itea3.org/project/web-of-objects.html.

[21] D. Guinard, V. Trifa, T. Pham, and O. Liechti, "Towards physical mashups in the web of things," in *Proceedings of 2009 6th International Conference on Networked Sensing Systems (INSS),* Pittsburgh, PA, 2009, pp. 1-4.

[22] The Internet-of-Things Architecture (IoT-A) Project [Online]. Available: http://open-platforms.eu/standard_protocol/iot-a-architectural-reference-model/.

[23] ITEA3 Projects, "DiY Smart Experiences: DIYSE D3.1 Service Oncology," [Online]. Available: https://itea3.org/project/diy-smart-experiences.html.

[24] BUTLER Project [Online]. Available: http://open-platforms.eu/standard_protocol/iot-a-architectural-reference-model/.

[25] iCORE Project [Online]. Available: http://www.iot-icore.eu/about-icore.

[26] W. Tushar, C. Yuen, K. Li, K. L. Wood, Z. Wei, and L. Xiang, "Design of cloud-connected IoT system for smart buildings on energy management," *EAI Endorsed Transactions on Industrial Networks and Intelligent Systems*, vol. 3, no. 6, pp. 1-9, 2016.

[27] S. K. Viswanath, C. Yuen, W. Tushar, W. T. Li, C. K. Wen, K. Hu, and X. Liu, "System design of the Internet of Things for residential smart grid," *IEEE Wireless Communications*, vol. 23, no. 5, pp. 90-98, 2016.

[28] T. Bylander, "The computational complexity of propositional STRIPS planning," *Artificial Intelligence*, vol. 69, no. 1-2, pp. 165-204, 1994.

[29] M. C. Koval, N. S. Pollard, and S. S. Srinivasa, "Pre-and post-contact policy decomposition for planar contact manipulation under uncertainty," *The International Journal of Robotics Research*, vol. 35, no. 1-3, pp. 244-264, 2016.

[30] R. B. Lamine, R. B. Jemaa, and I. A. B. Amor, "Graph planning based composition for adaptable semantic web services," *Procedia Computer Science*, vol. 112, pp. 358-368, 2017.

[31] A. L. Blum and M. L. Furst, "Fast planning through planning graph analysis," *Artificial Intelligence*, vol. 90, no. 1-2, pp. 281-300, 1997.

Sheik Mohammad Mostakim Fattah https://orcid.org/0000-0002-9103-6089

He was born in Dhaka, Bangladesh, in 1990. He was a research assistant in Advance Networking Lab and recieved Master's degree in Computer and Information Communication Engineering at Hankuk University of Foreign Studies, Korea at 2016. He received his B.Sc. degree in Computer Science and Engineering from University of Dhaka, Bangladesh in 2013. His current research interests include Internet of things, web of objects, semantic web, knowledge based system, ubiquitous computing.

Ilyoung Chong https://orcid.org/0000-0001-8291-7530

He joined at the Electronics and Telecommunications Research Institute (ETRI) in 1980, and since 1996 he has served with professor at Hankuk University of Foreign Studies (HUFS), Korea. He has taken the role of Dean of Engineering College and Vice-President of HUFS from 2006 to 2012. He received the Ph.D. of computer science from the University of Massachusetts, USA 1992. He involved in research on communications service platform, service-oriented networking and IoT services. He has taken the chief of editor in Springer, Germany from 2008 to 2011. He has taken the role of project leader of Korean Consortium for two ITEA3 EU projects, Web of Objects (WoO) and EmoSpaces (Enhanced Affective Wellbeing based on Emotion Technologies for adapting IoT spaces) since 2011. He has taken editors to develop 7 ITU-T Recommendations on NGN, IPTV, Virtual Home Network and IoT. Now he is developing 3 draft Recommendations on affective computing service and data interoperability issues in data processing and management.