Computer Science and Information Systems 13(2):427–452 DOI: 10.2298/CSIS150701010H

Development of Middleware Architecture to Realize Context-Aware Service in Smart Home Environment

Hyun-Wook Kim[1], M. Robiul Hoque[1], Hyungyu Seo[1], and Sung-Hyun Yang[1]

[1]Department of Electronic Engineering,
Kwangwoon University
Seoul, 139-701, Republic of Korea
E-mail: {khw, robiul, tjgusrb92, shyang}@kw.ac.kr

Abstract. A smart home provides automated services based on the context of the home environment and user activity. Context acquiring, processing, reasoning, and disseminating to the services are complex tasks for a context-aware system. An appropriate middleware architecture could handle such complexity. In this paper, we proposed a middleware architecture for a context-aware system in smart home environment. Here, the context is modeled based on ontology using Web Ontology Language (OWL). In addition, a profile applied improved rule-based reasoning algorithm is integrated into this middleware to infer high-level contexts from available low-level contexts. Experimental result shows that the middleware provides more accurate and faster reasoning outcome compare with the traditional rule-based reasoning method. Moreover, context-aware service is also selected using the rule-based algorithm, where the service can be extended easily by adding new service rules in the service rule base.

Keywords: context-aware, middleware, ontology, profile, reasoning, service, smart home.

1. Introduction

A smart home incorporates technology capable of sensing the home environment and provides appropriate services to users based on the context of the home environment and the users' activities and preferences. Therefore, the main aim of establishing a smart home is to provide the proper services to the right user, at the right time, in the right place, and on the right device based on the available contexts. The term 'context' refers to any information about the situation of an entity, where an entity is a person, place or object [1]. The services in a smart home should be aware of the contexts and automatically adapt according to the changing contexts; this is known as context-awareness. Therefore, context-awareness is one of the most important characteristics of a smart home.

Although many researchers have presented context-aware systems for different domains, the development of context-aware services in the home environment is still a complex and time-consuming task due to the lack of appropriate middleware architecture support. Appropriate middleware architecture for the smart home should provide support for most of the tasks involved in dealing with contexts such as acquiring context from various sources, performing context reasoning, and carrying out dissemination of the context to the service composer.

In this paper, we proposed a three-layered context-aware middleware architecture. The goals of this architecture are as follows:– 1) to support sensor abstraction that means hiding sensing details, 2) to support context information processing, and reasoning, and 3) to facilitate easy development of a context-aware service. The sensing and managing layer provides sensor abstraction, which facilitates sensing. Therefore, the services do not need to manage the low-level sensing details. The processing layer provides the facilities to generate the context from the sensed data, to infer a high-level context from the available low-level contexts, and to represent the context in a formal way. For inferring, the reasoning engine uses the profile applied improved rule-based reasoning algorithm. By adding new service rules to the service rule base, the service can easily be extended. The service composition layer composes the several services based on the context by using service rules. A set of services is developed to prove the effectiveness of this middleware architecture. For this middleware architecture, the context is modeled based on ontology, using Web Ontology Language (OWL) [2]. Moreover, fuzzy functions are used to ensure a more precise context.

The remainder of this paper is organized as follows. Section 2 provides a discussion on related works. In section 3, middleware architecture is presented in detail, and in section 4, the ontology-based context modeling is discussed. In section 5, the implementation of the proposed middleware architecture with experimental results is presented, and in section 6, we conclude this paper and indicate the scope of further research.

2. Related Works

Context-aware systems have attracted much attention from researchers in recent years. Several context-aware systems have been developed to demonstrate the usefulness of this technology. The ContextToolkit [3] provides a framework and a number of reusable components by using an object-oriented approach. While it supports the rapid prototyping of sensor based context-aware applications, it does not provide a common context model for context knowledge sharing and reasoning. In the CoBrA [4] project, the authors proposed an agent-oriented infrastructure for context representation, sharing knowledge and the user’s privacy control. GAIA [5] is a distributed middleware for context awareness and semantic interoperability, in which context ontology is represented and is written in DAMLC OIL. CORTEX [6] is a context-aware middleware for pervasive and ad-hoc mobile environments based on concepts for autonomous sentient objects and component frameworks. CARISMA [7] uses the reflection paradigm to enhance the development of adaptive and context-aware mobile applications. MiddleWhere [8] is a distributed middleware architecture for location awareness. This platform allows the addition of different types of sensing technologies and determines the location information quality. SOCAM [9] is an ontology based, service oriented context-aware middleware architecture for developing context-aware services. This platform supports semantic representation, context reasoning, and context-knowledge sharing. EXEHDAUC [10] is another context-aware system, but it tends to approach the concept of context from data rather than by defining and modeling context independently.

In a context-aware system, reasoning is primarily used to deduce new knowledge from available facts, which is another major challenge in the context-aware system. Rule-based reasoning is a popular reasoning method used to build a context-aware system. In [11], the authors use First-Order Logic (FOL) rules to reason about context for building context-aware mobile services. In the semantic space architecture, two modules are used for retrieving and deriving new information from the OWL Knowledge Base (KB) [12]. The context reasoner enables the users to deduce higher level knowledge, based on the context data of the KB, using FOL rules. The same approach is also taken in the prototype context-aware implementation described in [13]. In each of these research works, rule-based reasoning has been combined with ontology. However, insufficient personalized context is a common issue for most rule-based context-aware systems because it requires a lot of personalized rules to apply the user's preferences. In other word, a separate rule is necessary to consider each preference for reasoning, which increases number of rules. To address this problem, profile can be a good alternative source of user preferences to infer personalized context. The profile represents the user preferences for context and service in a formal way. By using user profile, the system can become aware of the user choices in particular context, and computes more accurate high-level context. As a result, high quality context-aware services are also provided by the system. As the user preferences are provided by the profile, the system can infer all possible high-level contexts using only basic rules which are applicable for all users in a domain.

In contrast, AmbieSense [14] uses Case Based Reasoning (CBR) to derive the current situation from available sensed data. When new information becomes available, the CBR agent attempts to retrieve a known context or case, and classifies the current situation based on the retrieved case. A similar approach is also implemented in [15] and [16]. The Chaining CBR (CCBR) approach [17] has also been presented to solve the imprecision of the traditional CBR. The CCBR considers that similar problems have similar solutions, and that solutions for similar prior problems are a useful starting point for new problem-solving. However, these methods still comprise the challenges of the representation of cases, management of cases in a context-aware system, case matching algorithm for finding similar cases, and dealing with personalized contexts.

In other research, attempts were made to adopt a user profile more specifically to assist users to shop for various items [18]. However, the purpose of this is to calibrate the quality of a service rather than to recognize the user context more precisely. Nevertheless, we proposed a middleware architecture that integrates, firstly, the functionalities of sensor abstraction to hide the low-level sensing details, and secondly, the functionalities of context processing with the help of ontology-based context modeling, fuzzy function, and a profile applied improved rule-based reasoning algorithm, and finally, the functionalities of the service selection and the composition technique to provide services for the smart home. Profile-applied improved rule-based reasoning uses the profiles while providing reasoning for the contexts, and provides a better outcome with which to infer the personalized context over traditional rule-based reasoning.

3. Middleware Architecture

Our proposed middleware architecture consists of several functional components, which are the software modules as shown in Fig. 1. The modules are related to each other and organized in a layered fashion. Moreover, within a module, several sub-modules work together to deliver the overall functionality of this architecture. This middleware architecture consists of three layers: the sensing and managing layer, the processing layer, and the service composition layer. In the following subsection, each part of this middleware architecture is briefly described.

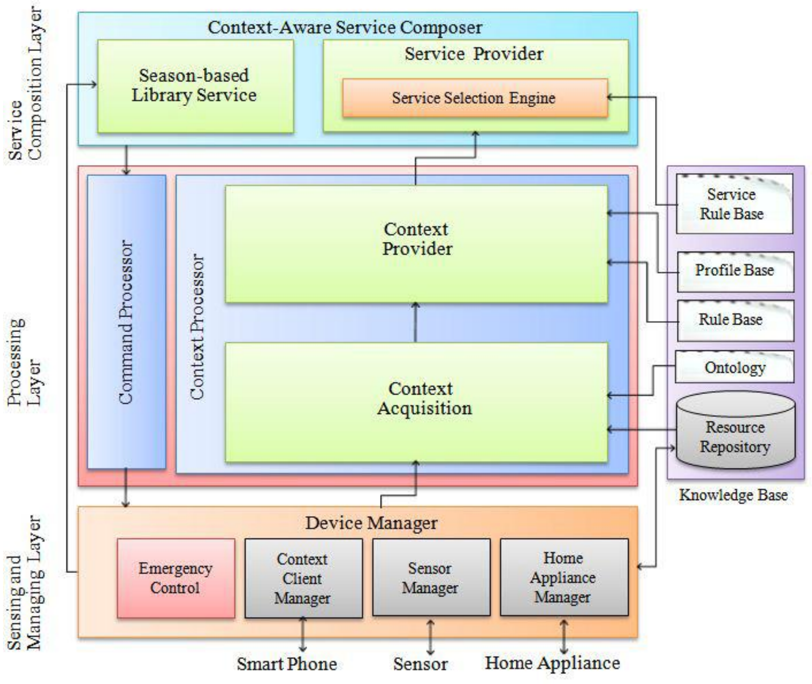


Fig. 1. Proposed middleware architecture

3.1. Sensing and Managing Layer

The sensing and managing layer controls and maintains all devices related to the smart home. Several devices such as sensors, home appliances, and smart phones are connected with the residential gateway via a heterogeneous home network. The device manager, in this layer, is a collection of several sub-modules that perform mainly the sensing and managing tasks.

Sensor Manager A sensor device detects various types of inputs (light, heat, motion, pressure etc) from the physical environment and transmits electronically over a network for further processing. In our smart home, we use wireless sensors which are connected to a sink node using Zigbee protocol, whereas the residential gateway communicates to the sink node using TCP/IP protocol. TCP/IP communication protocol allows connecting large number of devices which are not limited with distance. Therefore, using TCP/IP communication protocol, it is possible to connect a large number of sensors and appliances in smart home as well as other domain. RFID tag is used to identify a user's location in home. The sensor manager checks the sensor status (e.g. sensor normal, sensor failure, power failure, and poor communication) and reads data from the sensor based on an event such as when a user enters a new location in the home; otherwise, the sensor manager reads the sensor data repetitively according to a predefined time interval. It then passes on these data to interested components through EventBus[1] with a predefined format as shown in Table 1. Since the home environment parameters change slowly, to prevent similar data processing repetitively, the Euclidean distance based similarity checking method is used to discard similar data.

Home Appliance Manager. This module mainly controls the statuses of home appliances by setting up its control parameters according to the home appliance command for a particular service. In this case, the controller of the home appliances is communicated to the residential gateway using TCP/IP protocol, and the home appliances are wirelessly connected to the controller using IR/RF.

Context Client Manager. The context client manager plays a similar role as that of the sensor manager but receives data from the embedded sensors of the context client (e.g. smart phone) and sends them to the next layer for further processing. It also executes the service command to provide a service. Here, the context clients and the residential gateway communicate with each other wirelessly (e.g. WiFi) using TCP/IP protocol.

[1] EventBus is selected from the Google Guava library, which allows publish-subscribe style communication between software modules without requiring the modules to explicitly register with each other. It's like broadcasting; more than one module can receive simultaneously. Moreover, it makes the code much more clear and modularize.

Emergency Control. Emergency control is a special module that provides an emergency service. We developed this module as a special case to respond quickly in emergency situations. An emergency situation might be a fire (smoke sensor value is true), a house breaking (a true value of magnetic sensor indicates intrusion), and high CO_2 level (greater than the threshold). When the emergency control detects an emergency situation based on the sensing data, it provides an emergency message to the appropriate user or agency.

3.2. Processing Layer

The processing layer has three major parts: context processor, command processor, and knowledge base. The context processor is responsible for the acquisition of sensed data and provides context by processing those data. On the other hand, the command processor provides an interface between the sensing and managing layer, and the service composition layer. The knowledge base stores the necessary information that is required for the middleware architecture.

The context processor has two main sub-modules: context acquisition and context provider. Moreover, each sub-module has several further sub-modules, each of which is briefly described below.

Context Acquisition. The context acquisition module receives the sensed data from the device manager and makes low-level contexts using the ontology. Fig. 2 shows a detailed diagram of the context acquisition module. Several functions of this module are described in the following subsections.

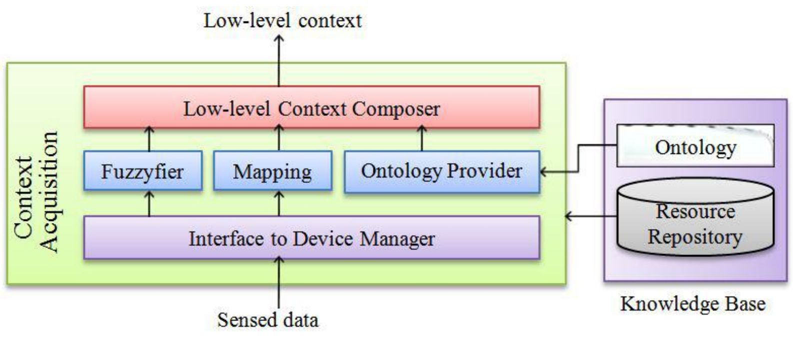


Fig. 2. Context Acquisition module in detail

Interface to Device Manager. This module provides protocol to receive data from the sensing and managing layer. First, this module receives data with a predefined format, then decomposes these data and sends them to the next module for further processing.

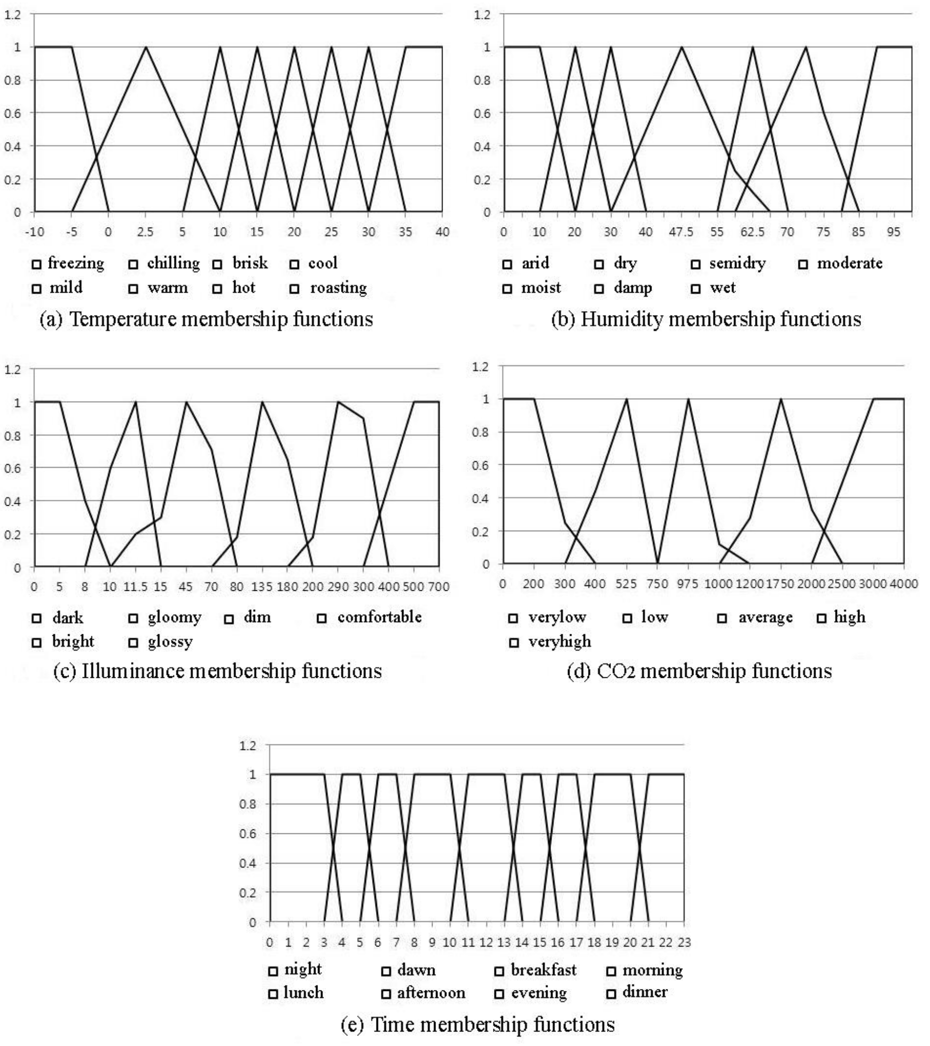
The data format consists of a flag, a tracker ID or location node ID, and a data value, as shown in Table 1.

Table 1. Sample data according to the data format

Data Format	Description
{“L”,“3”,“1”}	Flag “L” indicates user location, “3” is tracker id indicates who the user is, and “1” is location node id of *bedroom1*. That means, some user with tracker id 3, is located in *bedroom1*.
{“E”,“1”,“25.6”,“22”,“12”,“440”}	Here “E” indicates environmental information and “1” in second position is location node id of *bedroom1*. From 3rd position, environmental information are placed such as temperature, humidity, illuminance, and CO_2 respectively of this space.
{“P”,“1”,“1”}	Bed Pressure in *bedroom1* is ON.

Fuzzyfier/Mapping. The mapping module provides mapping between the sensed data and the instance of context entities which is needed in the subsequent steps of the middleware, such as mapping of the location node ID to the location name. On other hand, fuzzy functions are invoked to make more precise linguistic context information from the data of the residential group (temperature, humidity, CO_2, and illuminance) sensor, and time. The main reason for using fuzzy functions is to convert the sensed numeric data to linguistic expression, because most of the time people use the linguistic expression rather than numbers when they feel concerned about the external environment. For example, to express their understanding of temperature, people use phrases such as cool, hot, and so on, instead of referring to the temperature in ^{0}C. However, the range of external environmental information varies from person to person. Therefore, it is wise to use fuzzy sets to generate more realistic linguistic expressions from vague ranges. To fuzzify the sensing data, we defined fuzzy sets and membership functions as shown in Fig. 3.

Low-level Context Composer. The low-level context composer generates both existential and relational low-level contexts and represents them in a formal way as described in subsection 4.1. This module receives context instances from the fuzzyfier and/or the mapping module, and finds concepts of these instances from the ontology. The existential context is composed using the instance and its concept, for example, *bedroom(bedroom1)*. To make the relational context, the low-level context composer uses the ontology provider to retrieve the relation between the concepts of the related instances from the ontology. In ontology, the relation between concepts is predefined. Then, this module makes the relational context with these context instances and their relationship. For example, *locatedin(Mr.Hong,bedroom1)*, where locatedin is a relation between a person and the bedroom concepts, and *Mr.Hong* and *bedroom1* are the instances of these concepts, respectively. The pseudo code of this process is depicted in algorithm 1.



(a) Temperature membership functions
(b) Humidity membership functions
(c) Illuminance membership functions
(d) CO2 membership functions
(e) Time membership functions

Fig. 3. Fuzzy sets and membership functions for (a) Temperature, (b) Humidity, (c) Illuminance, (d) CO_2, and (e) Time.

Ontology Provider. The ontology provider provides an interface to access and maintain the smart home ontology. Some functionalities of this module are open and save the ontology, remove and insert individuals, and search and retrieve the relations among concepts etc. To develop an ontology provider, the OWL API and Pellet [19] reasoner are used. A smart home ontology will be described briefly in section 4.

Algorithm 1. Low-level relational context composition
(Let *a* and *b* be the instances of two concepts; *a* and *b* are generated through the fuzzyfier/mapping module; *r* is the relation between the concepts of *a* and *b*.)

```
Context low-level-context-composer (string a, string
b){
  Retrieve concept of the individual a from ontology;
  Retrieve concept of the individual b from ontology;
  Retrieve relation, r between the concepts of
individual a and b;
  Make low-level context with the format, r(a,b);
  Return low-level context;
}
```

Context Provider. The context provider consists of several sub-modules, as shown in Fig. 4, which are responsible for producing high-level contexts using the reasoning engine and maintaining context consistency. An example of a high-level context is *hasstatus(Mr.Hong,sleeping)* or *hasstatus(Mr.Hong,watchingtv)*. Finally, all available contexts are conveyed to the service composition layer to create the services.

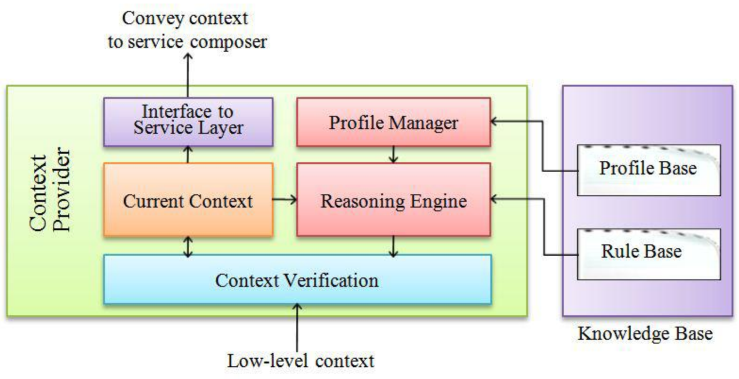


Fig. 4. Internal structure of context provider

Context Verification. This module checks whether or not a new context is available in the current context list. If it is available, the module simply discards this context; otherwise, it adds it to the current context list. If a new context is an updated version of a previous context, then it only modifies this context to change it to a new context. As a whole, the functions of this module are very important to prevent context duplication and maintain context consistency.

Current Context. Current context is the list of available contexts stored in the working memory. If any change is detected in the current context list, the context reasoning engine is invoked to compute high-level context from the available contexts. The context list is then sent to the service composition layer where the service is provided based on these contexts.

Reasoning Engine. The reasoning engine is a specially designed module which is used to infer new high-level context from the available current contexts. To develop this module, a user defined rule-based inferring technique is used. We integrated the user profile in the reasoning algorithm to infer a more accurate personalized context. The user profile provides the user's preferences and the rule is temporarily modified according to the profile to infer a personalized high-level context. This technique increases the context reasoning accuracy as well as decrease runtime.

The reasoning engine runs every time when a new context is added to the current context list or a context in the current context list is updated follow the sensed data. The reasoning engine also runs again if a new context is generated as a result of the reasoning, because it changes current context list. Theoretically, this process is intended to be repeated, therefore, it requires chaining algorithms such as the forward and backward chaining algorithms for its operation. As soon as a change is found in the current context list, the reasoning engine loads the list of rules from the rule base, and then starts to find matches from the current context list. Every rule contains variables, and each variable will be replaced by instances of some concepts. Context representation and its terminologies are described in section 4.1 in detail. We describe this reasoning process considering an example as follows.

In the rule for sleeping state, let *person(?p)^bedroom(?br)^ locatedin(?p,?br) ^hasilluminance(br,dark)* be defined as part of the rule. Here, the variable '?p' is mapped to the person and the variable '?br' is mapped to the bedroom. Therefore, in the case of *Mr.Hong* with his location in *bedroom1*, the rule is modified temporarily to *person(Mr.Hong) ^bedroom(bedroom1) ^locatedin(Mr.Hong,bedroom1) ^hasilluminance(bedroom1,dark)* by substituting variables with corresponding instances.

Once the substitution is completed but the rule is not satisfied with current context list, the reasoning engine finds the user who is related to the modified rule. Then the user profile of the corresponding user is loaded and the instances or values of the contexts are replaced according to the user's preference. A form of the further modified rule, in this case, is *person(Mr.Hong)^ bedroom(bedroom1) ^locatedin(Mr.Hong,bedroom1) ^hasilluminance(bedroom1,dim)*. Because, for the user Mr.Hong, Fig. 5 shows that one of the choice on illuminance is dim. After the modification has been completed, the reasoning engine again start to find the exact contexts that are the same as each modified condition in the rule and available in the current context list. That means, it finds whether the further modified rule is satisfied or not with current context list. Whenever the rule is satisfied, a high-level context is created and sent to the context verification. The context verification then adds it to the current context list after checking context consistency. The pseudo code of the main part of the reasoning algorithm is depicted in Algorithm 2. If a match is not found, the reasoning engine checks whether or not the high-level context is still valid. The user status is changed to *none*, when the condition for the high-level context is failed. More detailed description of the reasoning engine with preliminary testing results can be found in [20].

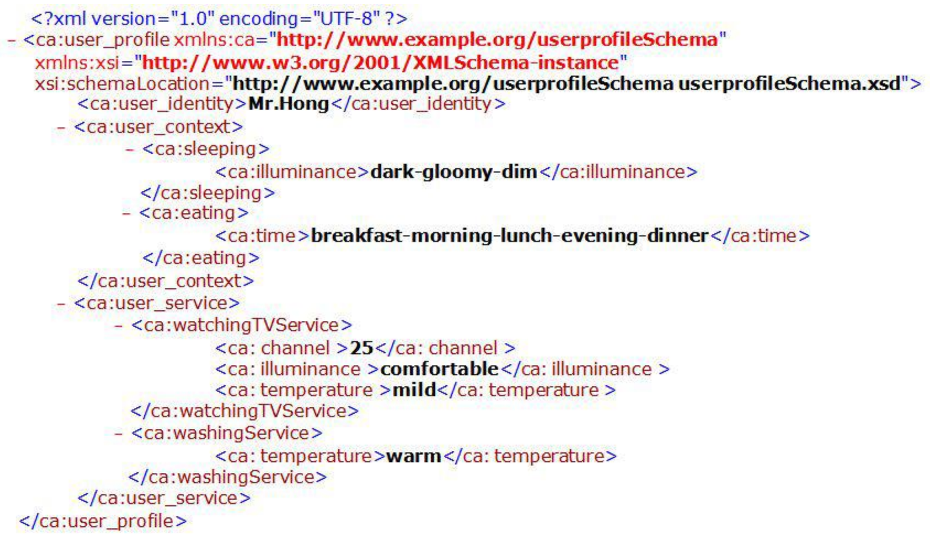
Algorithm 2. Profile applied improved rule-based reasoning

(This is a simplified pseudo code of the algorithm in which-

- An instance is an individual of the concept in the domain. For example, Mr. Hong is an instance of a concept person.
- A variable represents any instance of a concept. For example, person(?p), here ?p is a variable which can be mapped to any person in the domain, i.e. Mr. Hong, Mr. Kim, etc.
- The profile defines the user's preference for each context. For example, Fig. 5 shows that Mr. Hong's preferences for illuminance are dark, gloomy, and dim.
- A rule is a production rule similar in First-Order Prediction Calculus (FOPC). Several contexts with variables are combined by $\wedge$ to form a rule.
- `ruleVariable` is a list of all types of variables in a rule.
- `currentVariable` is the current processing variable of the ruleVariable, pointed by keyIndex.)

```
int keyIndex=0;
Boolean findMatch(int keyIndex){
  matched:=false;
  currentVariable:=ruleVariable[keyIndex];
  if(currentVariable is the last variable of
  ruleVariable){
    if(currentVariable has no instances)
      return false;
    else{
      while(currentVariable has instances){
        Substitute rule's currentVariables with the
        instance;
        if(!matches(rule with current context)){
          Retrieve preference on that instance from
          profile;
          Modify the rule according to preference;
          }
        if(matches(rule with current context)){
          Add if part of the rule to matched rule list;
          Add the then part to current context list
          through context verification;
          matched:=true;
          }
        }
      }
    }
  else{
    if(currentVariable has no instances)
      return false;
    else{
      while(currentVariable has instances){
        Substitute rule's currentVariables with the
        instance;
        keyIndex++;
        if(findMatch(keyIndex)){
          matched:=true;
          keyIndex--;
          }
        else
          keyIndex--;
        }
      }
    }
  return matched;
  }
```

Profile Manager. This module retrieves and applies profiles in the reasoning process. The profile defines the user’s preference for each context. Using the profile in reasoning, the resulting context as well as the ordinary services can be affected; however, in previous researches, the profile was only utilized for supporting personalized services [18], [21], [22]. When applying the profile, a number of rules can be reduced to compute the personalized user context. In this research, the profile is written using Extensible Markup Language (XML), an example of which is shown in Fig. 5.



```
 <?xml version="1.0" encoding="UTF-8" ?>
- <ca:user_profile xmlns:ca="http://www.example.org/userprofileSchema"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.example.org/userprofileSchema userprofileSchema.xsd">
      <ca:user_identity>Mr.Hong</ca:user_identity>
    - <ca:user_context>
        - <ca:sleeping>
                <ca:illuminance>dark-gloomy-dim</ca:illuminance>
          </ca:sleeping>
        - <ca:eating>
                <ca:time>breakfast-morning-lunch-evening-dinner</ca:time>
          </ca:eating>
      </ca:user_context>
    - <ca:user_service>
        - <ca:watchingTVService>
                <ca: channel >25</ca: channel >
                <ca: illuminance >comfortable</ca: illuminance >
                <ca: temperature >mild</ca: temperature >
          </ca:watchingTVService>
        - <ca:washingService>
                <ca: temperature>warm</ca: temperature>
          </ca:washingService>
      </ca:user_service>
 </ca:user_profile>
```

Fig. 5. XML definition of a user profile

Interface to Service. The interface to service module continuously checks the current context list, if any change is detected then conveys all contexts to the service composition layer using EventBus.

Command Processor. The command processor provides an interface between the service composition layer and the sensing and managing layer. This module receives a service command from the context-aware service composer and sets command parameters for the device manager according to the service command. Finally, the device manager generates a data packet based on command parameters to execute this service. Hence, the service command is a set of information needed to provide a particular service, such as device ID, device serial number, device status, functioning level, and location.

Knowledge Base. The knowledge base stores the domain ontology, rule base, and profile base in separate files. In addition, the home structure information, user information, and sensing data are stored in a database. Here, the resource repository is the main database, responsible for storing different types of information which are related to the smart home. It maintains several tables to store information about user, user’s location, sensors, smart phones, and home appliances. The database is implemented using MySql with JDBC connector. Rule files are stored as .txt file, whereas profiles are stored as .xml and ontology files are stored as .owl.

3.3. Service Composition Layer

In the service composition layer, the context-aware service composer allows various services based on available contexts which are received from the context provider and service rules from the service rule base.

Season-based Library Service. The season-based library service module is designed to provide a number of season-based special services. Currently, two services are provided by this module: temperature control and humidity control. When the temperature and humidity values are changed to very high or very low in a particular season, the sensor manager module directly invokes this module to generate environment control services. These services are important to maintain the home environment at a satisfactory level of comfort to live, even though the user is outside or the status is *none*. This module makes a decision based on sensing data. For example, if the room temperature is greater than 30^0C in the summer season, it generates a service command to regulate the air-conditioner automatically. As these are the default services, we develop this module separately to provide these services without undergoing the complex context processing steps.

Service Provider. The service provider module is responsible for providing services by following available contexts. It receives contexts from the context provider and selects the desired service using the service selection engine and service rule base. These services include sleeping, morning call, washing, watching TV, and so on. After selecting the appropriate service, this module composes the appropriate service command to achieve that service.

Service Selection Engine. The service selection engine is a rule-based inference engine. The role of this module is to infer the desired service by using the user define rules from the service rule base. The service selection engine uses a forward chain algorithm for rule-based inferring.

4. Context Modeling Using Ontology

In this model, context is defined as a collection of information that can represent the situation of any entity including the entity itself, its attributes, and its relation with other entities in the same domain from a specific perspective.

4.1. Context Representation

The context model affects the architecture of the system and its reasoning mechanism. If the context model is formal and structured, then the reasoning engine must provide formal context as the result of reasoning with formal contexts [23]. In this research, the context model is designed using Description Logic (DL). DL is a well-known knowledge representation language, which originated from First-Order Predicate Calculus (FOPC).

To present the context, this model uses terms such as concept, instance, relation, value, and object which are similar to the terms used in both DL and FOPC. The terms also use meanings closer to those in the Web Ontology Language (OWL). In OWL, the concept is a collection of the same types of instances, also called class. Instance, similar to the individual in DL, is an instantiated unique entity obtained by the concept in the domain. A relation is a link between two entities or between an entity and the attribute that defines the relationship which represents how one instance interacts with another. Moreover, value is literally a numeric or string representation of a subject's state.

Context is classified in this model as i) Existential Context (EC) and ii) Relational Context (RC). The EC is used to describe the existence of an entity in a specific domain, and its format is shown below.

Concept(Instance name)

As soon as the EC is created, it is stored in the current context list and will never be changed during runtime except time context. For example, *person(Mr.Hong)*. On the other hand, the RC literally demonstrates a connection between two entities. The RC is displayed as shown below.

Relation(Subject, Object or Value)

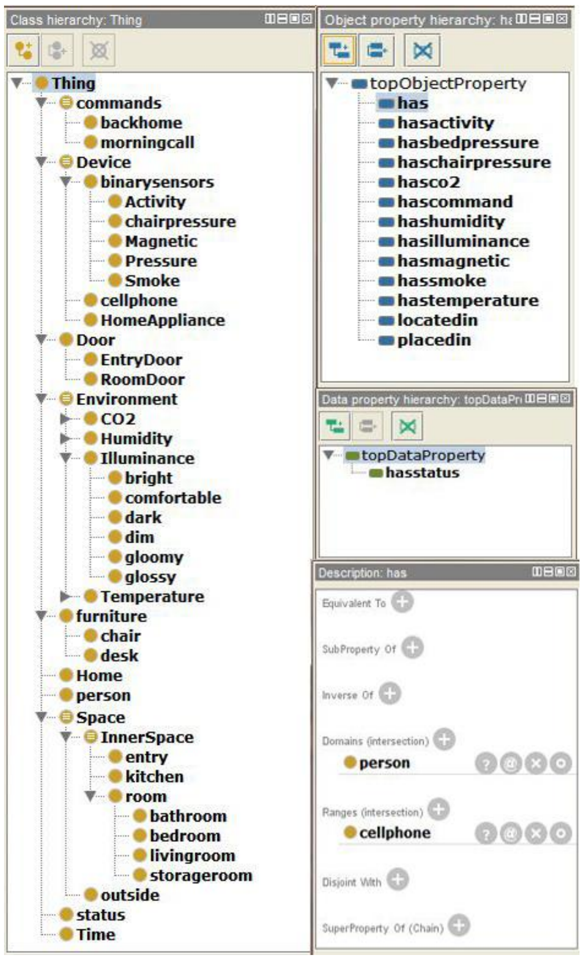
Each RC should contain one relation, one subject, and either one object or value. For example, *locatedin(Mr.Hong,bedroom1)*, or *hasstatus(Mr.Hong,Sleeping)*.

Previously, well known classifications of context were low-level context and high-level context. The low-level context is directly detected from the sensed data. On the other hand, high-level context is derived from low-level contexts by the reasoning process. In this model, the relational context can be either a low-level or high-level context, but an existential context is always a low-level context.

4.2. Smart Home Ontology

Context modeling is the specification of all entities and the relations between these entities, which are needed to describe the context as a whole. Several methods are available for context modeling: key-value models, mark-up scheme models, graphical models, object oriented models, logic based models, and ontology based models [24]. Key-value models use simple key-value pairs to define the list of attributes and their values describing the context information. Markup modeling is a hierarchical data structure consisting of markup tags with attributes and content. The main drawbacks of these approaches concern their limited capabilities in capturing a variety of context types; capturing relationships, dependencies, and quality of context information; allowing consistency checking; and supporting reasoning on context [25]. Graphical models have a strong graphical presentation and they are used in particular limited applications such as a relational database in an information system. Object oriented models encapsulate context processing at an object level and hence are hidden to other components; on the other hand, all logic based models are maintained at a high degree of formality. Therefore, these models have less expressiveness and interoperability. In the Ref [24], the authors conclude that ontology is a promising instrument for modeling contexts among other models. Ontology is a formal explicit definition of a shared conceptualization and it has many benefits; for example, a common ontology enables knowledge sharing, ontologies with well defined declarative semantics provide logic inference about contextual information, and explicitly represented ontologies allow knowledge reuse [4],[26]. Many ontology languages are available, such as RDF(S), OIL, DAML+OIL, and OWL for publishing and sharing ontologies. Moreover, several user-friendly graphical tools are available (e.g. Protégé) that make the design of ontology based context models also viable to developers who are not particularly familiar with description logics. These advantages have made ontology based system the main trend of context-aware systems among non-probabilistic approaches [27], [28]. Thus, ontology is a good candidate to express context and domain knowledge.

In this model, home entities are mainly categorized into person, device, space, environment, furniture, and time classes. Each class can be divided into several subclasses. For example, space emphasizes the spatial aspects of a smart home domain. Space has two subclasses: inner space and outside. Again, the inner spaces of the home domain are divided into room, kitchen, and entry. Furthermore, the room is divided into bedroom, living room, bathroom, and storage room. Fig. 6 shows a partial view of class hierarchy, object properties, data type properties, and the constraints that are defined in this model using Protégé editor.



```
Class hierarchy: Thing
Thing
  commands
    backhome
    morningcall
  Device
    binarysensors
      Activity
      chairpressure
      Magnetic
      Pressure
      Smoke
    cellphone
    HomeAppliance
  Door
    EntryDoor
    RoomDoor
  Environment
    CO2
    Humidity
    Illuminance
      bright
      comfortable
      dark
      dim
      gloomy
      glossy
    Temperature
  furniture
    chair
    desk
  Home
  person
  Space
    InnerSpace
      entry
      kitchen
      room
        bathroom
        bedroom
        livingroom
        storageroom
    outside
  status
  Time
```

```
Object property hierarchy: h
topObjectProperty
  has
  hasactivity
  hasbedpressure
  haschairpressure
  hasco2
  hascommand
  hashumidity
  hasilluminance
  hasmagnetic
  hassmoke
  hastemperature
  locatedin
  placedin
```

```
Data property hierarchy: topDataPr
topDataProperty
  hasstatus
```

```
Description: has
Equivalent To
SubProperty Of
Inverse Of
Domains (intersection)
  person
Ranges (intersection)
  cellphone
Disjoint With
SuperProperty Of (Chain)
```

Fig. 6. Class hierarchy, properties, and constraints of the model

5. Implementation of Middleware Architecture

The test-bed smart home, for this research, consists of the two bedrooms, a living room, a bathroom, and a kitchen. The smart home is fitted with a number of sensors grouped into residential, security, and human sensors. Residential group sensors are the temperature, humidity, CO_2, and illuminance sensors. Intrusion (door open) and fire (smoke) detection sensors are grouped within the security sensor group. User identification (RFID Tag), user movement detection (PIR sensor), and user's presence on the bed or chair (pressure sensor) detection sensors are organized in the human sensor group. Home appliances (e.g. air conditioner, DVD, TV, refrigerator, curtain, water heater, etc) are placed in different positions in the home. Sensors, home appliances, home server, and smart phones are connected to the residential gateway using both wired and wireless networks.

This middleware architecture has been implemented using the Java programming language. The resource repository is stored in the home server in which we need to register i) all family members' information including name, contact number, birthday, and age, ii) home structures information which includes bedrooms, living room, kitchen, windows, doors, and other home spaces information, and iii) a list of home appliances, which includes information about appliance type, serial number (representing different devices of the same type), and location of these devices. The components of the middleware architecture are installed on the residential gateway which processes all types of domain context information and controls the behavior of the home devices.

To observe the performances of this architecture, we consider a selective scenario based on the normal daily life in the test-bed smart home. In this scenario, initially, the user wakes up at the morning according to the alarm setting. Then he goes to the *bathroom* for taking shower. After finishing the shower and dressing up, he moves to kitchen for meal and then leaves for school. After school, he returns home and enters *bedroom2* to prepare his homework. He then enters the *living room* and watches TV sitting on the sofa. Finally, at night, he enters *bedroom1*, turns off the light, and lies on the bed to sleep. Although the light is off, the room is not completely dark because some streetlights are located close to the windows. However, he sleeps and the activities are repeated in the scenario for every day.

5.1. Context Reasoning Result

The context reasoning results from the traditional rule-based method and from the profile-applied improved rule-based method are compared. We choose the sleeping state to bring out user's preference for context. To compare the performances of the two reasoning methods, consider the user, Mr. Hong, prefers the illuminance of *dark*, *gloomy*, and *dim* for sleeping, that means he can sleep in the dark as well as lighter environment. The sensing data and its transformation for the considering sleeping state is given in Table 2. Data is converted to the context by the low-level context composer and added to the current context list.

Table 2. Sensed data and corresponding low-level contexts

Scenario	Sensed data packet	Added low-level context in current context list
Sleeping	E 1 25.2 50.1 15 200	hastemperature(bedroom1,warm), hashumidity(bedroom1,moderate), hasilluminance(bedroom1,dim), hasco2(bedroom1,verylow)
	L 3 1	locatedin(Mr.Hong,bedroom1)
	P 1 1	hasbedpressure(bedroom1,on)

A set of user defined rules is provided before booting up the system. A number of existential contexts are also added in the current context list as soon as the system has been booted up. Some of these rules are listed in Table 3. The partial current context list for a particular time is presented in Table 4 with the reasoning result. In rule number 1 of Table 3, the value of illuminance is set as *dark* for sleeping, but Table 4 shows the illuminance value in *bedroom1* as being *dim*. Using the profile-applied reasoning method, the inferred fact is *hasstatus(Mr.Hong,sleeping)* because the user profile states that one of Mr. Hong's preferences is *dim* for illuminance. On the other hand, the traditional method failed to catch the user preference, so it infers *hasstatus(Mr.Hong,none)*. In this way, applying a profile to the different context shows a significant promising reasoning result superior to the traditional method.

Table 3. Partial list of user defined rules

Rules for inferring new facts
1. if person(?p)^bedroom(?br)^locatedin(?p,?br)^hasilluminance(?br,dark) ^hasbedpressure(?br,on)^ hasstatus(?p,none) then hasstatus(?p,sleeping)
2. if person(?p)^bedroom(?br)^hasbedpressure(?br,on)^cellphone(?c)^has(?p,?c)^ hascommand(?c, morningcall)^locatedin (?p,?br)^hasstatus(?p,sleeping) then hasstatus(?p,wakeup)
3. if person(?p)^livingroom(?lr)^locatedin(?p,?lr)^ tv(?t)^haspower(?t,on) ^haschairpressure(?lr,on)^hasstatus(?p,none) then hastatus(?p,watchingtv)
4. if person(?p)^bathroom(?bt)^locatedin(?p,?bt)^hasstatus(?p,none) then hasstatus(?p,washing)
5. if person(?p)^kitchen(?k)^locatedin(?p,?k)^haschairpressure(?k,on)^ time(breakfast)^hasstatus(?p,none) then hasstatus(?p, eating)

5.2. Service Providing Result

In this system, we implemented a rule-based service selection engine to deduce the desired service using a forward chain algorithm. The service rules are presented in a similar way to the reasoning rules. Table 5 shows a partial list of service rules. We developed several context-aware services in a smart home domain. The environment control service controls the temperature and humidity in the smart home. The guarding service ensures security, while the comfort service controls the heating, cooling, humidity, and ventilation in the smart home. To facilitate personal daily life activities, morning call, washing, studying, watching TV, sleeping, and dining services are developed. Some of these services are illustrated in Table 6. These services also ensure efficient use of resources in the smart home. Services based on the above mentioned scenario are demonstrated several times and the middleware architecture shows an acceptable performance on average.

Table 4. Comparison of inferring results

	Profile applied rule-based reasoning	Traditional rule-based reasoning

Current context list	person(Mr.Kim)
	cellphone(01049387435)
	person(Mr.Hong)
	cellphone(01027423445)
	has(Mr.Kim,01049387435)
	has(Mr.Hong,01027423445)
	hasstatus(Mr.Kim,none)
	hasstatus(Mr.Hong,sleeping)
	livingroom(livingroom)
	entry(entry)
	bathroom(bathroom)
	kitchen(kitchen)
	bedroom(bedroom1)
	bedroom(bedroom2)
	outside(outside)
	desk(desk1)
	chair(sofa1)
	placedin(desk1,bedroom2)
	placedin(sofa1,livingroom)
	hastemperature(bedroom1,warm)
	hashumidity(bedroom1,moderate)
	hasilluminance(bedroom1,dim)
	hasco2(bedroom1,verylow)
	locatedin(Mr.Hong,bedroom1)
	hasbedpressure(bedroom1,on)
		person(Mr.Kim)
		cellphone(01049387435)
		person(Mr.Hong)
		cellphone(01027423445)
		has(Mr.Kim,01049387435)
		has(Mr.Hong,01027423445)
		hasstatus(Mr.Kim,none)
		hasstatus(Mr.Hong,none)
		livingroom(livingroom)
		entry(entry)
		bathroom(bathroom)
		kitchen(kitchen)
		bedroom(bedroom1)
		bedroom(bedroom2)
		outside(outside)
		desk(desk1)
		chair(sofa1)
		placedin(desk1,bedroom2)
		placedin(sofa1,livingroom)
		hastemperature(bedroom1,warm)
		hashumidity(bedroom1,moderate)
		hasilluminance(bedroom1,dim)
		hasco2(bedroom1,verylow)
		locatedin(Mr.Hong,bedroom1)
		hasbedpressure(bedroom1,on)

User status	Mr. Hong: sleeping	Mr. Hong: none

Table 5. Partial list of service selection rules

Service	Rules
Morning Call Service	if person(?x) ∧ bedroom(?y) ∧ locatedin(?x,?y) ∧ hasstatus(?x, sleeping) ∧ cellphone(?z) ∧ has(?x ?z) ∧ alarmcall(?z) then morningCallService(?x,?y)
Washing Service	if person(?p)^locatedin(?p,?bt)^ hasstatus(?p,washing) then washingService(?p,?bt)
Watching TV Service	if person(?p)^locatedin(?p,?lr)^hasstatus(?p, watchingtv) then watchingTVService(?p,?lr)

5.3. Performances Analysis

Our middleware architecture receives sensed data, generates context, infers high-level context and provides proper services to user. We evaluated proposed middleware architecture in terms of context reasoning outcome, service providing outcome and runtime of context reasoning. Experiment results presented in section 5.1 show that the proposed profile applied reasoning algorithm has better outcomes for inferring personalized context than the traditional algorithm. With the better inferring outcome, our middleware provides different services accurately to the user, which is also described in section 5.2. The function proposed in Algorithm 2 stretches the depth recursively when considering different variables and backs off when the modified rule is false. This characteristic limits the depth of this function, which is proportional to the number of variables, thus restricting the required size of memory. Therefore, it uses a limited memory space that is more valuable to make embedded system in the small domain like smart home.

Table 6. Illustration of some services with corresponding important functions

Service	Description of functionalities
Morning Call Service	Open window's curtain, play music on DVD player
Washing Service	Turn on bathroom light and regulate water temperature based on user preference.
Watching TV Service	Adjust living room light, temperature, and suggest TV channel according to the user's choice.

Applying the profile, a number of rules can be reduced to catch the user's preferences in context because the system needs only basic rules that are applicable for all users. This means that only a single rule is required to infer a single status for all users, although they have different preferences in different contexts. On the other hand, for the same outcome, the traditional algorithm needs as many as rules, equal to the different preferences in the different contexts of the basic rule for different users, which increases its runtime. Therefore, it is obvious that the proposed algorithm needs fewer rules as well as less runtime, as shown in Table 7.

Table 7. Comparison of time complexity between proposed and traditional algorithm

Description	Specification	
	Proposed algorithm	Traditional algorithm
Number of possible outcomes of a rule	$\prod_{i=1}^{m} P(x<\infty)_i$	$\prod_{i=1}^{m} P(x<\infty)_i$
Worst time for inference	$O(n \times m)$	$O(n \times m \times d)$
Best time for inference	$\Omega(m)$	$\Omega(m \times d)$

Where,
-n is the number of contexts in current context list
-m is the number of variables in a rule
-x is the number of instances for each variable
-d is the total number of different preferences of different users for variables m

Sensor nodes of our smart home are connected to a sink node wirelessly using Zigbee protocol, whereas the residential gateway communicates to the sink node using TCP/IP protocol. The sensor manager reads data from the sensors when an event occurs, such as, when a user enters in a new location in the home; otherwise, the sensor manager reads the sensor data repetitively according to a predefined time interval, if there is a significant change in the sensing data. For testing the performance of these algorithms, a volunteer user acted twelve different activities in the test-bed smart home and taken the record of his activities manually. During the user's activity period, the middleware collects sensing data and stores in the database automatically. Sensed data is collected for a period of four weeks. The total number of data packets per day depends on number of event occurs, repetitive sensing time interval, and change in sensing data. A major time, user stays outside for works in day and sleeps in night. So, user movement in home is limited. Moreover, as environmental parameters in home change slowly, a number of similar sensed data will be discarded. However, the number of data packets stored in the database daily is around 250 to 300 with normal user movement and 10 minutes time interval.

We run both proposed and traditional reasoning algorithm using stored data packets for inferring high-level context. The runtime for each algorithm is measured in several times. The average runtime for both algorithms with respect to the number of data packets of a day are plotted in Fig. 7. This figure shows that the runtime of both algorithms are proportionally increased to the number of data packets. In case of the traditional algorithm, the runtime is increased more rapidly than the proposed algorithm. These reveal that our algorithm performs better than the traditional algorithm when number of data packets is increased. Experiment result shows the proposed algorithm takes at average 38% less runtime than the traditional algorithm.

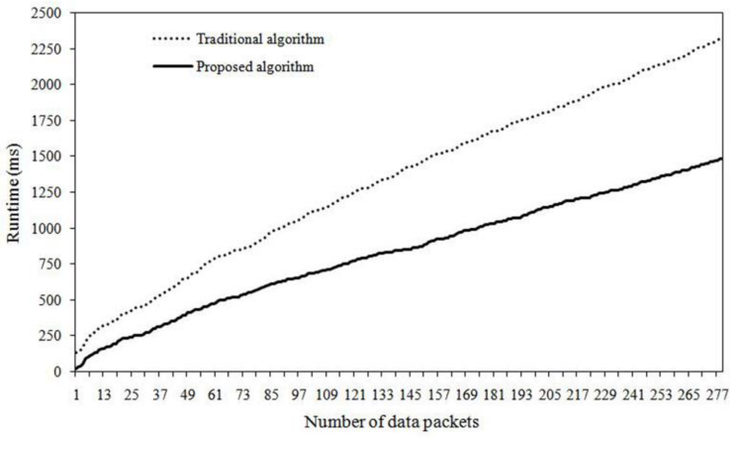


Fig. 7. Comparison of runtime between proposed and traditional reasoning algorithm

We also measure context reasoning runtime based on the number of user preferences in contexts. The Fig. 8 shows that the runtime of the proposed algorithm is almost same with respect to increasing number of preferences. But the runtime of traditional rule-based algorithm is increased by a faster rate.

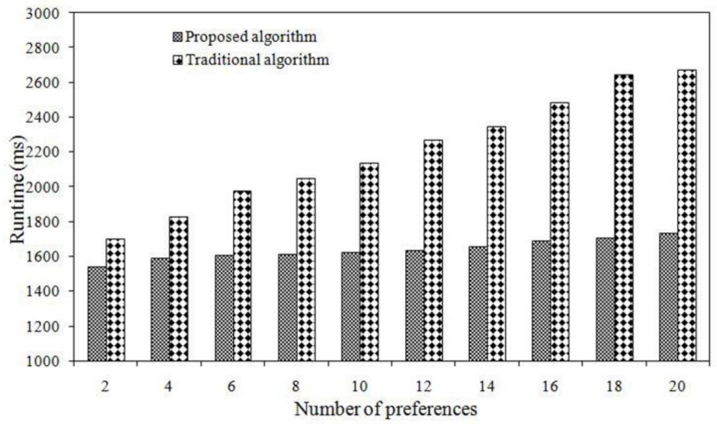


Fig. 8. Comparison of runtime with respect to the number of preferences in contexts

By applying user profile, the proposed algorithm provides same outcome using less number of rules compared with traditional algorithm. In both algorithms, the corresponding rule set is repetitively reevaluated for inferring high-level context whenever a new context is generated either from new data packet or by reasoning. The number of evaluated rules for both algorithms with respect to number of activities is depicted in Fig. 9. The figure shows that the proposed algorithm evaluates fewer rules than the traditional algorithm with fixed number of data packets.

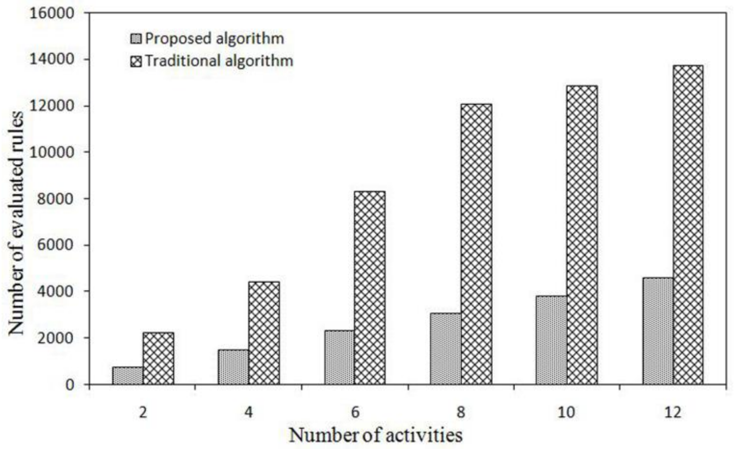


Fig. 9. Comparison of evaluated rules with respect to the number of activities

Therefore, the proposed algorithm always provides better result in case of accuracy and runtime using less number of rules; as a result, the proposed middleware can provide service more accurately and instantly.

6. Conclusions

In this research, we presented a comprehensive middleware architecture design and its implementation issues. To facilitate middleware functionalities, we explained formal context modeling using OWL for the smart home domain. The middleware utilizes the smart home ontology to obtain semantic information such as the relation and concept of existing instances in the domain. Moreover, we introduced a new profile applied reasoning algorithm which uses profiles while inferring high-level context. It shows better performance over traditional algorithm to reason personalized context. The proposed method is validated by comparing its reasoning outcome with manually collected user states. This shows reliable reasoning accuracy in smart home domain based on normal daily life activities. The inferring algorithm is developed using the forward chaining method which tests all possible outcomes recursively, and the depth of this algorithm is restricted by the number of variables. Therefore, this algorithm occupies less memory. In addition, as the user profile provides user preferences in contexts, the proposed algorithm needs reduced set of rules and consequently the runtime is also reduced significantly. On the other hand, with the improved reasoning accuracy, this middleware always recommends the suitable service for the user.

We faced several challenges during the implementation and testing of the proposed middleware architecture. Sometimes, the users perform more than one activity at a time, or they perform cooperative or shared activities. It is challenging to recognize these types of complex states. At present, we consider only a single state at a time. To recognize multistate for a single user or multi users at a time in same place, specialized sensor networks with wearable sensors, image sensors, etc. are needed. In addition, user’s preferences may change over time. Therefore, to provide a more reliable context as well as service, the profiles need to be maintained and kept up to date. A machine learning technique can be applied to update profile from history data. Besides that, erroneous sensing data can lead the wrong context reasoning as well as wrong service selection. Erroneous sensing data can be occurred due to the malfunction of sensor network. A preprocessing method need to apply on sensing data to avoid this type of error. Moreover, we consider limited number of contexts and implement our present test-bed smart home with selected number of simple daily life activities. We have to consider more contexts and complex activities to develop real smart home.

In the future, we will upgrade the proposed middleware architecture by inserting different context reasoning techniques and incorporating a machine learning algorithm for inferring ambiguous contexts to support those challenges. For a machine learning algorithm, we will consider reinforcement learning because it is able to enhance the function by accepting feedback.

Acknowledgement. This work was supported by the Industrial Strategic Technology Development Program (10041788, Development of Smart Home Service based on Advanced Context-Awareness) funded by the Ministry of Trade, Industry & Energy (MI, Korea). The present research has been conducted by the Research Grant of Kwangwoon University in 2016.

References

1. Dey, A. K. and Abowd, G. D.: Towards a Better Understanding of Context and Context-Awareness. In Proceedings of the 1st International Symposium on Hand-held and Ubiquitous Computing, 304-307. (1999)
2. Smith, M., Welty, C. and McGuinness, D.: Web Ontology Language (OWL) Guide. www.w3.org/TR/owl-guide/. (2004)
3. Dey, A. K., Salber, D. and Abowd, G. D.: A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. Human-Computer Interaction (HCI) Journal, Vol. 16(2), 97-166. (2001)
4. Chen, H., Finin, T. and Joshi, A.: An Ontology for Context Aware Pervasive Computing Environments. The Knowledge Engineering Review, Vol. 18(3), 197-207. (2003)
5. Ranganathan, A. and Campbell, R. H. : A Middleware for Context-Aware Agents in Ubiquitous Computing Environments. In Proceedings of ACM/IFIP/USENIX International Middleware Conference, Rio de Janeiro, Brazil, 143-161, (2003)
6. Blair, G. S., Coulson, G. and Grace, P.: Research Directions in Reflective Middleware: the Lancaster Experience. Proceedings of the 3rd workshop on Adaptive and reflective middleware, 262–267, (2004)
7. Capra, L., Emmerich, W. and Mascolo, C.: Carisma: Context-Aware Reflective Middleware System for Mobile Applications. IEEE Transactions on Software Engineering, Vol. 29, 929-945, (2003)
8. Ranganathan, A., Al-Muhtadi, J., Chetan, S., Campbell, R. and Mickunas, M. D.: Middlewhere: a Middleware for Location Awareness in Ubiquitous Computing Applications. In Proceedings of the 5th ACM/IFIP/USENIX international conference on Middleware, 397–416. (2004)
9. Gu, T., Pung H. K. and Zhang, D. Q.: A Service-Oriented Middleware for Building Context-Aware Services. Journal of Network and Computer Applications, Vol. 28(1), 1-18. (2005)
10. Lopes, J., Gusmao, M., Souza, R., Davet, P., Souza, A., Costa, C., Barbosa, J., Pernas, A., Yamin, A. and Geyer, C.:Toward a distributed architecture for context-aware mobile applications in UbiComp. In Proceedings of the 19th Brazilian symposium on Multimedia and the web, WebMedia, 43-50. (2013)
11. Gu, T., Pung, H.K., and Zhang, D.Q.: A Middleware for Building Context-Aware Mobile Services. In Proceedings of the IEEE Vehicular Technology Conference (VTC 2004), Milan, Italy, (2004)
12. Wang, X.H., Dong, J.S., Chin, C.Y., Hettiarachchi, S.R., and Zhang, D.: Semantic Space: an infrastructure for smart spaces. IEEE Pervasive Computing, Vol. 3(3), 32–39, (2004)
13. Ranganathan, A., and Campbell, R.H.: An infrastructure for context-awareness based on first order logic. Personal and Ubiquitous Computing, Vol. 7(6), 353–364, (2003)
14. Mikalsen, and M.Kofod-Petersen, A.: Representing and Reasoning about Context in a Mobile Environment. Revue d'Intelligence Artificielle, Vol. 19(3), 479-498, (2005)
15. Wiratunga, N., Craw, S., Taylor, B., and Davis, G.: Case-based reasoning for matching SMARTHOUSE technology to people's needs. Knowledge-based Systems, Vol. 17, 139-146, (2004)
16. Ma, T., Kim, Y.-D., Ma, Q., Tang, M., and Zhou, W.: Context-Aware Implementation based on CBR for Smart Home. IEEE International Conference on Wireless and Mobile Computing, Networking and Communications, Vol. 4, 112-115, (2005)
17. Nguyen, T. V., Woo, Y. C., and Choi, D.: CCBR: Chaining Case Based Reasoning in Context-Aware Smart Home. First Asian conference on intelligent information and database systems, 453-458. (2009)
18. Morikawa, D., Honjo, M., Yamaguchi, A. and Ohashi, M.: A proposal of user Profile Management Framework for context-aware service. In Proceedings of the Symposium on Applications and the Internet Workshops, 184-187. (2005)

19. Sirin, E., Parsia, B., Grau, B. C., Kalyanpur, A. and Katz, Y.: Pellet: A practical OWL-DL Reasoner. Journal of Web semantics: Science, Services and Agents on the World Wide Web, Vol. 5, 51-53. (2007)
20. Hoque, M. R., Kabir, M. H., Seo, H. and Yang, S.-H.: PARE: Profile-Applied Reasoning Engine for Context-Aware System. International Journal of Distributed Sensor Networks, Hindawi Publishing Corporation, Vol. 2016, 1-11. (2016)
21. Groppe, J. and Mueller, W.: Profile Management Technology for Smart Customizations in Private Home Applications. Iin Proceedings of the 16th International Workshop on Database and Expert Systems Application, 226-230. (2005)
22. Thomsen, J., Vanrompay, Y. and Berbers, Y.: Evolution of context-aware user profiles. In Proceedings of the International Conference on Ultra Modern Telecommunications and Workshops, 1-6. (2009)
23. Cho, E.-S., Yoon, T.-S., Choi, J.-H., Paik, J.-Y. and Helal, S.: An Integrated Formal Model for Context-Aware Systems. In Proceedings of the IEEE 37th Annual Computer Software and Applications Conference Workshops, 163-168. (2013)
24. Strang, T. and Linnhoff-Popien, C.: A context modeling survey. In Proceedings of the International Workshop on Advanced Modeling Reasoning and Management UbiComp. (2004)
25. Bettini, C., Brdiczka, O., Henricksen, k., Indulska, J., Nicklas, D., Ranganathan, A., and Riboni, D.: A survey of context modeling and reasoning techniques. Pervasive and Modeling Computing, Vol. 6(1), 161-180. (2010)
26. Fensel, D.: Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce. Springer. (2003)
27. Bulling, A. and Zander, T.O.: Cognition-Aware Computing. Pervasive Computing, IEEE, Vol. 13(3), 80-83. (2014)
28. Kim, J. Y. and Lee, G. M.: Context awareness for smart ubiquitous networks. In Proceedings of the International Conference on electronics, Information and Communications, 15-18. (2014)

Hyun-Wook Kim has received his Bachelor of Engineering degree in Computer Engineering from Kwangwoon University, Seoul, Republic of Korea in 2009. Now he is pursuing his unified course of the master’s and the doctor’s at same University. His main research interests are Embedded System, FPGA, Home Network and Sensor Network.

M. Robiul Hoque has received B.Sc (Hon’s) and M.Sc. degree in Computer Science and Engineering from Islamic University, Kushtia, Bangladesh, in 2003 and 2004 respectively. He is an Assistant Professor at same University. Now, he is pursuing his Doctoral degree at Kwangwoon University, Republic of Korea. His main research interests include Context-Aware System, Ubiquitous Computing, Smart Home, and Sensor Networks.

Hyungyu Seo has received his Bachelor of Engineering degree in Computer Engineering from Kwangwoon University, Seoul, Republic of Korea in 2014. Now he is pursuing his master’s degree at same University. His main research interests are Context-Aware System, Sensor Network and Embedded System.

Sung-Hyun Yang has completed his Ph.D. from Kwangwoon University, Seoul, Republic of Korea in 1993. He is a Professor in Electronic Engineering at same University. He is a Director of the Ubiquitous Home Network Center, Kwangwoon University. He was a Research Scientist at Boston University from 1996 to 1998. He was Chairman of the Home Network Market Activation Section, Korean Association for Smart Home from 2007 to 2008. His main research interests are Digital Logic, Embedded Systems, M2M, Next Generation Ubiquitous Home Networks, and Context-Aware System.

Received: July 01, 2015; Accepted: May 22, 2016