

A Context-aware Service Platform in a Smart Space *

Wan-rong Jih Li-lu Chen Jane Yung-jen Hsu
jih@agents.csie.ntu.edu.tw j94017@csie.ntu.edu.tw jyhsu@csie.ntu.edu.tw
Department of Computer Science and Information Engineering
National Taiwan University
Taiwan

ABSTRACT

Ubiquitous computing technology plays a key role for providing context information and makes context-aware services can be delivered in a smart space. As the contexts changing rapidly, context resources can be gathered from sensors, mobile devices, and personal information softwares and a context-aware system must be aware of such environment changes and provides adaptive and proactive services to the users. In addition, all the inner computing operations have to be hidden behind the users.

We propose a Context-aware Service Platform, implemented on JADE, and it utilizes Semantic Web technologies to analyze the ambient contexts and contrive service plan. We integrated ontology and rule-based reasoning to automatically infer high-level contexts and deduce a goal of context-aware services. An AI planner decomposes complex services and establishes the execution plan. Agents perform the specified task to accomplish the service. A *Smart Alarm Clock* scenario demonstrates the detail functions of each agent and shows how these agents incorporate with each others.

1. INTRODUCTION

The world's information infrastructure continues to fragment, and various information can be collected by using tiny, battery-powered, and low-cost mobile computer devices, such as PDAs, smart phones, and wireless sensors. Utilize the information of a physical environment, for example, temperature, humidity, monitoring light or any other environmental factor, can provide more intelligent and adaptive services to users.

In a smart space, augmented appliances, stationary computers, and mobile sensors provide raw context information, and a context-aware system must understand the meaning of the context, that is, a way to represent context is our first issue.

*This research was supported by the Innovative and Prospective Technologies of Institute for Information Industry under contract number (95)-1086.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

AAMAS'07 May 14–18 2007, Honolulu, Hawai'i, USA.

Copyright 2007 ACM International Workshop on Agent-Based Ubiquitous Computing ...\$5.00.

A smart space should have capability for inferring the raw context to high-level context. For instance, what is the activity in the room? Where is the user? Context-aware services require the high level description about the user's state and the environmental condition. How to infer such higher level context is the second issue.

Services are built on different platforms and run on their original computers without needing to migrate applications to personal devices. Each service must use exactly the standard communication mechanism for communicating with others. Transmission of service requests and the results are sent via wireless networks, such as WiFi. Furthermore, different applications are concerned with different types of knowledge and their corresponding representation models, whereas the messages between services should be understood by each other. Therefore, how to handle such knowledge sharing and maintain the consistency of knowledge is another issue.

This research explores the roles of intelligent sensing, wireless communicating, mobile and ubiquitous computing in smart home services for users. We introduce the context-aware service platform, which provides context-aware services to the user resident in a intelligent space.

Interaction between the user and services in the smart space through a wide variety of appliances for data gathering and information presentation. Services of the environment tracks the location and specific activities of the user through sensors, such as pressure-sensitive seats, bed sensors, infrared remote controller, and smart gadgets. Meanwhile, the user receives multimedia messages or content through speakers and monitors, as well as smart home devices. *Context-aware computing* enables the services in the intelligent environment to respond, at the right time and in the right place, to the user's needs based on the collected sensor data. The Context-aware Service Platform is the solution and provides the implementation architecture to achieve the goal for fully supporting the demands of the user's daily life in the smart space.

The rest of this paper is organized as follows: Section 2 discusses the technologies to build the Context-aware Service Platform. The infrastructure of this platform is introduced in Section 3 and the next section concentrates on the detail of each functional component. Section 5 demonstrates a *Smart Alarm Clock* scenario and its detailed design. Finally, Section 6 and 7 list the related work and a conclusion, respectively.

2. TECHNOLOGIES OVERVIEW

An overview of the context-aware systems, service-oriented

architectures, and context model are introduced in this section.

2.1 Context-Aware Systems

During the past years, a number of context-aware systems have been developed to support pervasive computing and ambient intelligent environments such as Active Badge location system[17], PARCTAB[16], and Context Toolkit[15]. These systems utilize various sensors and devices to provide location-aware services but lack knowledge sharing and context reasoning.

Typical researches dealing with context reasoning are Easy Meeting[6] and MyCampus e-Wallet[9]. EasyMeeting is a prototype of an intelligent meeting room that built on a Context Broker Architecture (CoBrA), an agent-based broker that maintains all the context knowledge represented in RDF-triple and utilizes Jena and Jess to support context reasoning. Besides, the policy of SOUPA (Standard Ontology for Ubiquitous and Pervasive Applications)[5] is used to control the sharing of users' contextual information. The MyCampus at Carnegie Mellon University has been developed to provide context-aware mobile services for the community in the university and the e-Wallet is its key element that supports access control and obfuscation rules for user's privacy preferences. Both EasyMeeting and MyCampus deployed context reasoning on their systems but they did not consider the needs of service integration.

2.2 Web Services

The concept of service integration can be constructed in a service-oriented architecture (SOA), in which loosely-coupled architecture services communicate with each other. Generally, such communication involves either simple message passing or services which coordinate some activities. Web services technology enables the connection of services.

W3C defines several Web server related specifications. Simple Object Access Protocol (SOAP)¹ for exchanging structured information in a decentralized, distributed environment. Web Service Definition Language (WSDL)² is an XML format for describing network services and the way how to access them. OWL-based Web Service Ontology, OWL-S³ is an ontology of services that makes software can discover, invoke, compose, and monitor Web resources. OASIS Technical Committees define the Universal Description, Discovery and Integration (UDDI)⁴ which forms the necessary technical foundation for publication and discovery of Web services implementations both within and between enterprises.

To make service integration, a framework needs to describe a standard ontology for declaring and sharing services. A reasoning application can be built into the framework, so that it can automatically determine the logical consequences of ontologies.

2.3 Context Models

In order to know the changes of environment, researchers usually define context models to represent the context information. Typically, time and location are the most well-known context models. Temporal reasoning plays an es-

sential role in context-aware systems. DAML time ontology[10] and ISO 8601 date and time formats[2] are the popular structure and standard. Two important temporal models are point-based and interval-based time models. Traditional time structure is based on a set of points, Bry *et al.*[4] introduce CaTTs and treat the cultural calendars as interval-based time. Ma and Hayes[12] analyze the temporal interval-based models in a recent literature report.

Many context-aware systems concentrate on location aware services. Maryland Information and Network Dynamics Lab Semantic Web Agents Project (mindswap) Group[14] develops Semantic geoStuff to express basic geographic features such as countries, cities, and relationships between these spatial descriptors. The Open Geospatial Consortium, Inc. (OGC)[13] prescribes OpenGIS specifications for GIS data exchange and process, and OpenCyc Spatial Relations[7] specify the vocabularies of spatial objects and relations.

3. SMART SPACE INFRASTRUCTURE

Figure 1 shows the infrastructure of a Context-aware Service Platform in a smart space. Context resources can be obtained from the computing softwares, such as personal calendar, weather forecasts, location tracking system, personal friend list, and shopping list, as well as raw sensor data. Context Collection Agents obtain raw contexts from softwares and hardware sensors and base on the context model, and then raw data are converted into a semantic representation. A Context-aware Service Platform is continuous collecting these contexts and infers appropriate service applications, then automatically and proactively delivers the services to users.

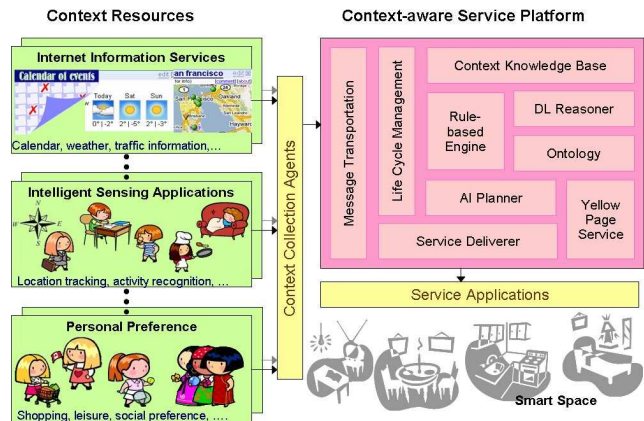


Figure 1: A Smart Space Infrastructure

A *Context-aware Service Platform* contains the following components:

Message Transportation provides a well-defined protocol for maintaining a set of communicative acts. Moreover, a common message structure is defined to exchange messages over the Context-aware Service Platform. Common message structure contains sender, receiver, the type of the communicative act, message content, and description of content. In this platform, each component communicates with each other through message passing.

¹<http://www.w3.org/TR/soap/>

²<http://www.w3.org/TR/wsdl>

³<http://www.w3.org/Submission/OWL-S/>

⁴<http://www.uddi.org/specification.html>

Life Cycle Management maintains a White Pages and states of services to control over access to and use of the services. A service can be in one of the following states: initiated, active, waiting, suspended, and deleted state. Life Cycle Management reflects the state changes and controls the state transitions. Consequently, every component is controlled by life cycle management.

Rule-based Engine uses IF-THEN rule statements, which are simply patterns and the inference engine performs the process of matching the new or existing facts against rules. Similar to DL reasoners, rule engines can also deduce high-level contexts from low-level contexts; the major difference is, rule engines can handle complex reasoning (*e.g.* combining several contexts to deduce higher-level contexts) while the DL reasoners can not. The derived high-level contexts are asserted into **Context Knowledge Base** which serves as a persistent storage for context information. Rules for which and when the appropriate service can be invoked are defined as the knowledge of service invocation rules for the Rule-based Engine.

Ontologies are loaded into the **DL reasoner** to deduce high-level context from low-level context. DL reasoner provides the inference services to ensure the ontology does not contain contradictory facts. The class hierarchy of an ontology can be used to answer queries by checking the subclass relations between classes. Besides, computes the direct types of every ontology instance can support to find the specific class that an instance belongs to.

Yellow Pages Service provides the functions for service registration and discovery. New services register their services to Yellow Pages Service. Service Deliverer and other services can search the desired services and get the results.

AI Planner generates a service composition plan sequence which satisfies a given goal. **Service Deliverer** chooses a service to execution from the service candidate list which returns from Yellow Pages Service.

4. CONTEXT-AWARE SERVICE PLATFORM

The Foundation for Intelligent Physical Agents (FIPA⁵) develops computer software standards to promote the inter-operation of heterogeneous agents and the services that they can represent. The Java Agent Development Framework (JADE⁶) is a FIPA-compliant software framework for multi-agent systems, implemented in Java and comprised several agents. There are an Agent Management System (AMS) controls agents' life cycle and play the role of white pages service, Directory Facilitator (DF) provides yellow pages service to other agents, an Agent Communication Channel (ACC) is the agent which can provide the path for basic contact between agents, and an Agent Communication Language (ACL) has been specified for setting out the message formats, consists of encoding, semantics, and parameters.

Design of the Context-aware Service Platform shows in Figure 2. The top block depicts a smart space environment. It provides *Context Resources* and delivers *Context Services*

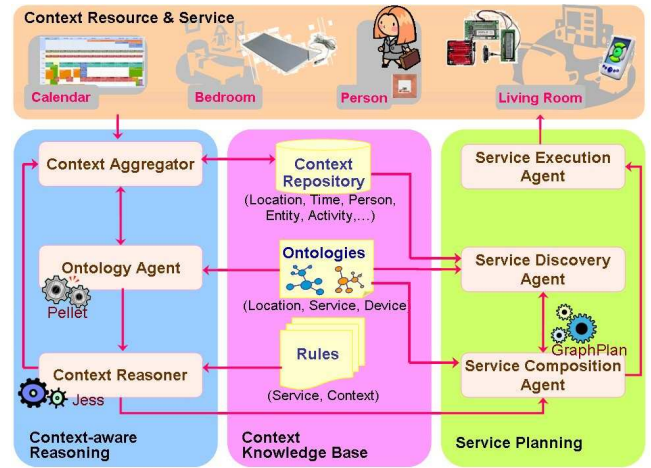


Figure 2: Functional Flow of Context-aware Service Platform

for the user. Contexts can be collected from sensors or software and will be delivered to *Context-aware Reasoning*. Ontology Agent and Context Reasoner can infer high-level context to provide service goal for Service Planning. Context information and service description are stored in *Context Knowledge Base*. After perform service composition, discovery, and delivery, a context-aware service will be delivered by *Server Planning*.

4.1 Context-aware Reasoning

We deploy Jess⁷ in our Context-aware Service Platform. Jess is a forward-chaining rule engine used Rete algorithm[8] to process rules, which is a very efficient algorithm for solving the difficult many-to-many matching problem. Moreover, an open-source OWL-DL reasoner Pellet⁸ developed by Mindswap Lab at University of Maryland and a Java framework for building Semantic Web applications Jena⁹ is used for providing a programmatic environment for RDF, RDFS, and OWL.

4.1.1 Context Aggregator

Initialization : A configuration file declares the type of context that the Context Aggregator will received and subscribe the context to its corresponding Context Collection Agent (refer to Figure 1).

Input message : There are two types of input contexts: (1) Raw context refers to data obtained directly from context sensors or software, such as bed sensor data and forecast data, can be delivered from a bed sensor and weather API respectively. Senders of these low-level contexts are called Context Collection Agents and the data will be wrapped as RDF-triple in message content. (2) High-level context is the information inferred from raw context, such as “location of a furniture” and “activity who currently participate in”, can be inferred from ontology reasoner and rule-based reasoner respectively.

⁵<http://www.fipa.org/>

⁶<http://jade.tilab.com/>

⁷<http://herzberg.ca.sandia.gov/>

⁸<http://pellet.owld.com/>

⁹<http://jena.sourceforge.net/>

Process : Value of a context can be changed at anytime and anywhere. Consequently, Context Aggregator must collect contexts and maintain the consistency of current context. Either raw or high-level context has an unique type identity and value. The associated value will be replaced when new context is arrived.

Output message : While raw contexts received from Context Collection Agents, the new context is immediately stored in Context Repository and a set of current context is delivered to Ontology Agent.

4.1.2 Ontology Agent

Initialization : An OWL context ontology describes structure and relation between contexts that will be loaded and parsed into RDF triples by Jena API. It also subscribes to Context Aggregator for the contexts that has been declared in the context ontology. In addition to ontology loading, it has to start a DL reasoner for supporting ontology query.

Input message : Context Aggregator sends the current state of contexts when any subscribed context has been changed.

Process : There are two types of ontology reasoning that perform in Ontology Agent, so they can provide high-level context. The first is inferred by Jena API that deduces high-level context from the object property of context, such as “bed sensor is attached to a bed” and “bed is placed in a bedroom”. Relationships between the instances of context object are defined in context ontology. A DL reasoner, *i.e.* Pellet, computes the inferred superclasses of a class and decides whether or not one class is subsumed by another, for example, “living room is an indoor location”.

Output message : If there is no high-level context to be derived, the input message of current contexts will be redirected to Context Reasoner. Otherwise, the new high-level contexts must be delivered to Context Aggregator.

4.1.3 Context Reasoner

Initialization : The Jess API provides packages to load a rule-based engine when Context Reasoner is started up.

Input message : A set of current context, which is the same as input message of Ontology Agent.

Process : The input context must be wrapped as Jess rule format and assert into the rule-based engine, and may trigger the execution of rules that can infer new contexts or derive a goal of service.

Output message : New high-level contexts are sent to Context Aggregator, whereas the service goal is delivered to Service Composition Agent.

4.2 Service Planning

The modern trend of software is to build a platform-independent architecture that distributes software components on the Internet. New services and functionalities can be automatically achieved by selecting and combining a set of available software components.

A service functionality contains a semantic annotation of what it does and a functional annotation of how it behaves. OWL-S (formerly DAML-S) is an ontology for services, and it provides three essential types of knowledge about a service: service profile, process model, and service grounding. An OWL-S service profile illustrates the preconditions required by the service and the expected effects that result from the execution of the service. A process model describes how services interact and how the functionality they offer can be integrated to provide a solution of the goal. The role of service grounding is to provide concrete details of message formats and protocols.

According to these semantic annotations, AI planning has been investigated for composing services. Graphplan[3] is a general purpose graph-based planner. The state transition is defined by operators consists of preconditions and effects. Given an initial state, goals, and operations, a planning system returns a service execution plan, which is a sequence of actions that starts from the initial state and accomplishes the given goals.

4.2.1 Service Composition Agent

Initialization : An OWL-S service ontology represents all available services and the service profile describes service goal, preconditions, and effects for AI planner, *i.e.* Graphplan. Consequently, the service ontology must be parsed and transferred into Graphplan operations.

Input message : A service goal is sent by Context Reasoner.

Process : According to the service operations, Graphplan creates a sequence of operation to achieve the goal.

Output message : When a operation represents a composite service, the corresponding service model will be delivered to Service Discovery Agent. If the execution plan has been generated, it delivers a sequence of service to Service Execution Agent.

4.2.2 Service Discovery Agent

Initialization : According to the description of the service model in service ontology, all atomic services will be kept in this agent.

Input message : A composite service model receives from Service Composition Agent.

Process : Given the composite service, Service Discovery Agent searches the atomic processes that are available and can carry out the composite service.

Output message : The atomic services, which can accomplish the given composite service, must be delivered to Service Composition Agent.

4.2.3 Service Execution Agent

Initialization : Service ontology consists of the description of service grounding, which specifies the details of how an agent can access a service.

Input message : A service list is sent by Service Composition Agent.

Process : Following the control sequence of services, the corresponding device agents will be invoked for providing atomic service.

Output message : The input parameters, invoking atomic service, must be passed to the device agent.

4.3 Context Knowledge Base

4.3.1 Context Repository

Context Repository contains a consistency of context, including location, time, person, and activity information. A RDF-triple represents contexts of the repository, like a subject, a predicate, and an object. Subject is a resource named by a URI with an optional anchor identity. The predicate is a property of the resource, and the object is the value of the property for the resource. The following triple represents “Peter is sleeping”.

```
<http://...#Peter>
<http://...#participatesIn>
<http://...#sleeping>
```

Where **Peter** represents subject, **participatesIn** is a predicate, and object **sleeping** is an activity. According to the elements of RDF-triple, we use subject and predicate as the compound key of Context Repository.

4.3.2 Ontologies

An ontology is a data model that represents a domain and is used to reason about the objects in that domain and their relations. We defines a context ontology depicts in Figure 3 as a representation of common concepts about the smart space environment. Context information are collected from

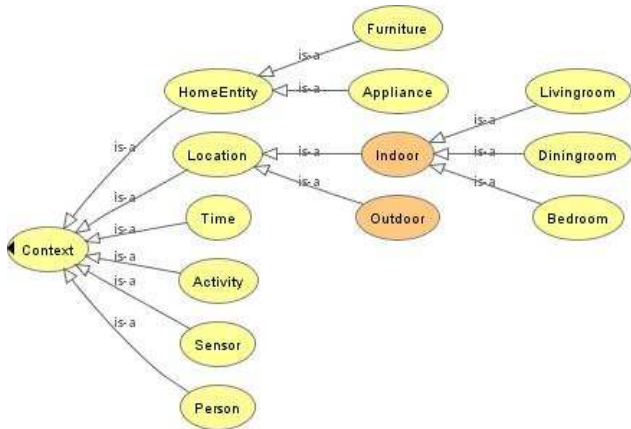


Figure 3: A Context Ontology

real-world classes (Person, Location, Sensor, Time, HomeEntry), and a conceptual class Activity. The class hierarchy represents an subclass relationship; an arrow points from a subclass to another superclass. The subclass relationship can declare whether the classes, belonging to the same superior class, are disjoint or not, for example, Indoor Location and Outdoor Location are two disjoint classes and both of them belong to Location class.

A service ontology defined by OWL-S is for describing available services that comprises service profile, service model, and service grounding, which has been stated in Section 4.2

4.3.3 Rules

Rules of a rule-based system serve as IF-THEN statements. Context rules can be triggered to infer high-level context. A rule, detecting whether a user is sleeping or not, is showed as follows:

```
(defrule User_is_sleeping
  (triple
    (subject ?person)
    (predicate "http://www.w3.org/...#type")
    (object "http://...#Person")
  )
  (triple
    (subject ?person)
    (predicate "http://...#isLocatedIn")
    (object "http://...#bedroom")
  )
  (triple
    (subject "http://...#bed_sensor")
    (predicate "http://...#isOn")
    (object "#true~http://...#boolean")
  )
  =>
  (assert
    (triple
      (subject ?person)
      (predicate "http://...#participatesIn")
      (object "#sleeping")
    )
  )
)
```

Patterns before ==> are the conditions, matched by a specific rule, called left hand side (LHS) of the rule. On the other hand, patterns after the ==> are the statements that may be fired, called right hand side (RHS) of the rule. If all the LHS conditions are matched, then the actions of RHS will be executed. The RHS statement can be either asserted new high-level contexts or delivered a service goal.

The **User_is_sleeping** rule shows that if ?person is a person, and ?person is in bedroom, and the bed sensor is on, then asserts the “?person is sleeping” as a fact.

5. A DEMONSTRATION SCENARIO

We use an example to illustrate detailed picture of Context-aware Service Platform.

In a smart space, a Smart Alarm Clock can check Peter’s schedule and set the alarm one hour prior to the daily first task. If Peter does not wake up within 5-minute period after the alarm is sent, send another sound of alarm and increases its volume. If Peter wake up early then the alarm time, there will be no more alarm.

On the other hand, when the first task event is approaching and the sensors detect that Peter remains sleeping, an exceptional control should deal with such situation.

In addition to send the alarm through traditional alarm clock, the alarm service can deliver to the devices that in Peter’s bedroom, such as radio, stereo, speaker, or personal mobile device.

5.1 Context-aware Reasoning

In order to archive *Smart Alarm Clock*, we have to collect Peter's schedule to decide the alarm time and should reason whether Peter is awake or not. Google Calendar Data API supports on-line schedule information and position-aware sensors, bed pressure sensors, etc., can detect whether user on the bed or not. RFID technologies can be used to recognize and identify the activities of Peter[19] while a wireless-based indoor location tracking system can determine Peter's location with room-level precision[18].

In order to know whether Peter is sleeping or not, all the related instances form an ontology instance network, shows in Figure 4. Dashed line indicates the connection of

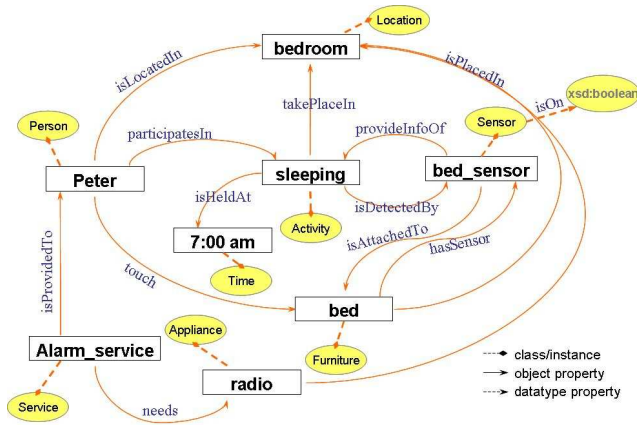


Figure 4: Instance Network about Sleeping

a class and its instance. Word in the box depicts a instance of its corresponding class, for example, bed is an instance of Furniture class. Each solid arrow reflects the direction of owl:ObjectProperty relationship, from domain to range and a inverse property can be declared while specifies the domain and range classes. For example, a sensor bed_sensor is attached to (isAttachedTo) furniture bed, and the inverse property is hasSensor. A boolean data type property isOn is associated with Sensor class for detecting whether the instances of class is on or off.

If someone is on the bed and the sensor bed_sensor is on, then the value of isOn is true. On the other hand, when nobody touches the bed, the value of isOn has to be false. While the bed_sensor is off, inferring that Peter is not sleeping, it is not necessary to deliver the Alarm_service.

Suppose that calendar agent reports the first event of the day will be held at 8:00am, therefore, the alarm is set to 7:00am. When the time is up, given the location of Peter and the status of bed sensor, the rule User_is_sleeping in Section 4.3.3 can deduce that whether Peter is sleeping or not. Assume that there is another rule reflects that "if Peter is sleeping, then deliver smart alarm service". Consequently, the service goal of *Smart Alarm Clock* can be derived and deliver to Service Composition Agent.

If Peter does not wake up for the task, according to the owner and importance of this task, an exceptional handling rules will be triggered for deciding whether to postpone or cancel this coming task. For instance, Peter has to host a meeting at 8:00am and hence the task is a high priority event. Consequently, a rule will be triggered to postpone the meeting, and an emergency message will be sent to the

corresponding participants for informing the situation. On the contrary, if this task is "watch TV show at 8:00am" with low priority, then the context-aware reasoning infers that this scheduled task should be canceled and a video recording event will be invoked.

5.2 Service Planning

Operations for planner can be derived from service profile, which gives a brief description about the service and consists of service name, preconditions, effects, inputs, and outputs of the service. The following OWL-S statements indicates the profile of *Smart Alarm Clock*.

```
<profile:Profile rdf:ID="alarm">
<profile:serviceName
  rdf:datatype="http://www.w3.org/...#string">
  smart alarm
</profile:serviceName>
<profile:hasPrecondition
  rdf:resource="#alarm_precond"/>
<profile:hasInput>
  <process:Input rdf:ID="message_stream">
    <process:parameterType rdf:datatype=
      "http://www.w3.org/...#anyURI">
      http://...#MessageStream
    </process:parameterType>
  </process:Input>
</profile:hasInput>
  ...
</profile:Profile>
```

This example shows a service named smart alarm has precondition alarm_precond and its input parameter belongs to MessageStream class.

Service model gives detailed description of the service and each service is modeled as process. There are three types of process: atomic process is the primary process without any subprocess, simple process are used as elements of abstraction, it can either represents as atomic process or composite process, and composite process consists of subprocesses. A composite process can be decomposed by using control operators such as sequence, split, split+join, choice, any order, if-then condition, iterate, repeat-until, and repeat-while. Figure 5 is the control flow for

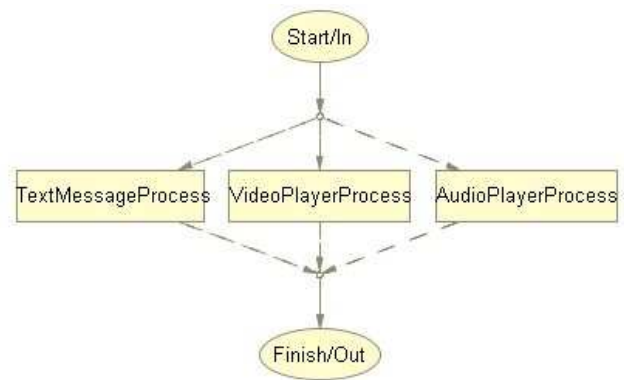


Figure 5: Process Graph of Smart Alarm Clock

Smart Alarm Clock, it uses operator choice to compose the process. The TextMessageProcess, VideoPlayerProcess,

and *AudioPlayerProcess* are atomic process, and *Smart Alarm Clock* can be served by using one of the three atomic processes. An example of *AudioPlayerProcess* shows as follows.

```
<process:AtomicProcess rdf:ID="AudioPlayerProcess">
  <process:hasInput>
    <process:Input rdf:ID="audio_stream">
      <process:parameterType
        rdf:datatype="http://...#anyURI">
          http://...#AudioStream
      </process:parameterType>
    </process:Input>
  </process:hasInput>
  <process:hasPrecondition>
    <expr:KIF-Condition rdf:ID="alarm_precond">
      <expr:VariableBinding
        rdf:ID="isSleepVariablebinding">
        <expr:theObject
          rdf:resource="http://...#sleeping"/>
        <expr:theVariable rdf:datatype="
          http://...#boolean"> true
        </expr:theVariable>
        </expr:VariableBinding>
      <expr:expressionData
        rdf:datatype="http://...#string">
        precondition of smart alarm
      </expr:expressionData>
    </expr:KIF-Condition>
  </process:hasPrecondition>
  <process:hasResult>
    <process:Result rdf:ID="alarm_done"/>
  </process:hasResult>
  .....
</process:AtomicProcess>
```

Descriptions of atomic process are similar with that of profile, except the service model describes process in more details. For example the input data type of *AudioPlayerProcess* is belong to *AudioStream* class, whereas the alarm service profile only gives an upper-level data type *MessageStream*. Moreover, atomic process describes detailed expression of preconditions, for instance, it binds an instance *sleeping* of *Activity* class to a boolean variable.

Service grounding specifies the details of the way to access the service, and deals with the concrete level of specification. Both OWL-S and WSDL are XML-based languages, therefore, the OWL-S service is easy to bind with WSDL service, for example:

```
<grounding:WsdLGrounding
  rdf:ID="AudioPlayerWSDLgrounding">
  <service:supportedBy
    rdf:resource="#AudioPlayer"/>
  <grounding:hasAtomicProcessGrounding>
    <grounding:WsdLAtomicProcessGrounding
      rdf:ID="WsdLAtomicProcessGrounding">
      <grounding:owlsProcess
        rdf:resource="#AudioPlayerProcess"/>
      <grounding:wsdlOperation>
        <grounding:operation
          rdf:datatype="http://...#anyURI">
            play
        </grounding:operation>
      <grounding:portType
```

```
        rdf:datatype="http://...#anyURI">
          audio player port type
        </grounding:portType>
      </grounding:wsdlOperation>
    </grounding:WsdLAtomicProcessGrounding>
  </grounding:hasAtomicProcessGrounding>
  .....
</grounding:WsdLGrounding>
```

A WSDL service has construction of *type*, *message*, *operation*, *port type*, *binding*, and *service*. The *AudioPlayerWSDL* grounding briefly shows that OWL-S has provided *operation* and *portType* mapping. Besides, the XSLT can help the transformation from WSDL descriptions to OWL-S parameters.

6. RELATED WORK

Smart spaces can be the homes, workplaces, cities, vehicles, and the spaces deploy embedded sensors, augmented appliances, stationary computers, and mobile devices to gather contexts of the user. Each place has different challenges, but similar technologies and design strategies can be applied. In order to make the space have capabilities to respond to the complexities of life, researchers explore new technologies, materials, and strategies to make the idea possible.

Department of Architecture research group at Massachusetts Institute of Technology proposes the *House_n*[11] research, which includes a “living laboratory” residential home research facility called the *PlaceLab*. Hundreds of sensing components are installed in nearly every part of the house. Interior conditions of the house can be captured by using these sensors, such as temperature, light, humidity, pressure, electrical current, water flow, and gas flow sensors. Eighty wired switches can detect the opening of the refrigerator, the shutting of the linen closet, or the lighting of a stovetop burner events. Cameras and microphones are embedded in the house for recording the resident’s movement. Twenty computers collect all the data streams from these devices and sensors to provide multi-disciplinary research, for example, monitoring the resident’s behavior, activity recognition, and dietary status. Besides, *Aware Home*[1] was proposed by the Future Computing Environments Group at Georgia Institute of Technology. In this house, multi-discipline sensors have been constructed for monitoring the activities of the resident.

These smart space projects didn’t organize the huge sensing data in a formal structured format. An independently developed application can’t easily interpret contexts that have no explicitly represented structure. We use the Semantic Web standards RDF and OWL to define context ontologies which provide a context model for supporting information exchange and interpret contexts. By using the Semantic Web technologies to represent context knowledge, we introduced an infrastructure for inferring higher-level contexts and provide adapt service to the user.

7. CONCLUSION AND FUTURE WORK

This paper presented Context-aware Service Platform, a prototype system designed to attain context-aware services in a smart space. This platform integrates several modern technologies, include ubiquitous and context-aware technologies, semantic web, AI planning, and web service. Be-

sides, reasoning approaches for deriving new contexts and services are adopted in this platform.

Ontologies for contexts and services provide information sharing and make the platform integrating services. Contexts are represented as RDF-triple for exchanging information between agents and deducing new high-level contexts. Moreover, the service planner obtains a goal from context-aware reasoner, such that it makes the services can be adaptively operated.

The current design assumed that all the context resources providing consistent contexts and no conflict information to disturb the process of Context-aware Service Platform. We should consider the fault tolerance problems but allow some minor errors happened.

As the real-world environment has huge number of contexts and the required tasks are much more complex, rule engine and Graphplan should have the capability to provide solutions in reasonable time. Consequently, the concept of clock timer can be adopted to the reasoning and planning components. In order to provide a possible solution from the partial results, an anytime algorithm should be taken into account.

This paper provides a simple scenario to demonstrate the idea of the context-aware service platform. However, this simple case does not show the power of automated service composition by using AI planning. Designing other scenarios that can explain the needs of service composition is one of our future direction. Applying this platform to other Web Service composition benchmark test is another way to evaluate the performance of this platform.

8. REFERENCES

- [1] G. D. Abowd, C. G. Atkeson, A. F. Bobick, I. A. Essa, B. MacIntyre, E. D. Mynatt, and T. E. Starner. Living laboratories: the future computing environments group at the georgia institute of technology. In *Proceedings of Conference on Human Factors in Computing Systems (CHI '00): extended abstracts on Human factors in computing systems*, pages 215–216, New York, NY, USA, 2000. ACM Press.
- [2] P. V. Biron and A. Malhotra. XML schema part 2: Datatypes second edition. <http://www.w3.org/TR/xmlschema-2/>, 28 October 2004. W3C Recommendation.
- [3] A. L. Blum and M. L. Furst. Fast planning through planning graph analysis. *Artificial Intelligence*, 90(1-2):281–300, February 1997.
- [4] F. Bry, F. A. Ries, and S. Spranger. CaTTS: calendar types and constraints for web applications. In *Proceedings of the 14th international conference on World Wide Web (WWW '05)*, pages 702–711, New York, NY, USA, 2005. ACM Press.
- [5] H. Chen, T. Finin, and A. Joshi. *Ontologies for Agents: Theory and Experiences*, chapter The SOUPA Ontology for Pervasive Computing, pages 233–258. Whitestein Series in Software Agent Technologies. Birkhäuser Basel, July 2005.
- [6] H. Chen, T. Finin, A. Joshi, L. Kagal, F. Perich, and D. Chakraborty. Intelligent agents meet the semantic web in smart spaces. *IEEE Internet Computing*, 8(6):69–79, November – December 2004.
- [7] Opencyc selected vocabulary and upper ontology – spatial relations. <http://www.cyc.com/cydoc/vocab/spatial-vocab.html>, December 2002.
- [8] C. L. Forgy. Rete: A fast algorithm for the many pattern/many object pattern match problem. *Artificial Intelligence*, 19(1):17 – 37, September 1982.
- [9] F. Gandon and N. M. Sadeh. A semantic e-wallet to reconcile privacy and context awareness. *Lecture Notes in Computer Science: The SemanticWeb - ISWC 2003*, 2870:385–401, October 20–23 2003.
- [10] J. R. Hobbs and F. Pan. An ontology of time for the semantic web. *ACM Transactions on Asian Language Information Processing (TALIP), Special Issue on Temporal Information Processing*, 3(1):66–85, March 2004.
- [11] S. S. Intille. Designing a home of the future. *IEEE Pervasive Computing*, 1(2):76–82, April-June 2002. House_n.
- [12] J. Ma and P. Hayes. Primitive intervals versus point-based intervals: Rivals or allies? *The Computer Journal*, 49(1):32–41, 2006.
- [13] OpenGIS specifications. <http://www.opengeospatial.org/standards/>, 2007.
- [14] F. Reitsma. Semantic geoStuff. <http://www.mindswap.org/2004/geo/geoStuff.shtml>, 2004. Maryland Information and Network Dynamics Lab Semantic Web Agents Project (mindswap).
- [15] D. Salber, A. K. Dey, R. J. Orr, and G. D. Abowd. Designing for ubiquitous computing: A case study in context sensing. Technical Report GIT-GVU-99-29, Georgia Institute of Technology, 1999.
- [16] B. Schilit, N. Adams, and R. Want. Context-aware computing applications. In *Proceedings of IEEE Workshop on Mobile Computing Systems and Applications*, pages 85 – 90, Santa Cruz, CA, US, 1994.
- [17] R. Want, A. Hopper, V. F. ao, and J. Gibbons. The active badge location system. *ACM Transactions on Information Systems (TOIS)*, 10(1):91–102, January 1992.
- [18] C. wen You, Y.-C. Chen, J.-R. Chiang, P. Huang, H. hua Chu, and S.-Y. Lau. Sensor-enhanced mobility prediction for energy-efficient localization. In *Proceedings of Third Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON 2006)*, volume 2, pages 565–574, Reston, VA, USA,, 2006.
- [19] C. yau Lin and Y. jen Hsu. IPARS: Intelligent portable activity recognition system via everyday objects, human movements, and activity duration. In *Modeling Others from Observations (MOO 2006): Papers from the 2006 AAAI Workshop*, number Technical Report WS-06-13, pages 44–52, Menlo Park, California, 2006. AAAI Press.