

Agent-based Context Consistency Management in Smart Space Environments

Wan-rong Jih
jih@agents.csie.ntu.edu.tw

Jane Yung-jen Hsu
yjhsu@csie.ntu.edu.tw

Han-Wen Chang
r96922005@ntu.edu.tw

Umi Laili Yuhana
yuhana@its-sby.edu

Department of Computer Science and Information Engineering
National Taiwan University
Taipei, Taiwan

Department of Informatics
Faculty of Information Technology
Institut Teknologi Sepuluh Nopember (ITS)

ABSTRACT

Context-aware systems in a smart space environment must be aware of the surrounding contexts and adapt to changing contexts in highly dynamic environments. Data managements of contextual information are different from traditional approaches because of the contextual information are dynamic, transient, and fallible. Consequently, capabilities to detect context inconsistency and maintain consistent contextual information are two key issues for managing contexts. We propose an ontology-based model for representing, deducing, and managing consistent contextual information. In addition, we use ontology reasoning to detect and resolve context inconsistency problems, which will be described in a *Smart Alarm Clock* scenario.

1. INTRODUCTION

It's obvious that mobile devices, such as smart phone, personal digital assistants (PDAs), and wireless sensors, are increasingly popular. Moreover, many tiny, battery-powered, and wireless-enabled devices have been deployed in smart spaces for collecting contextual information of residents. Customized information can be delivered across mobile devices, based on specific contexts (location, time, environment, *etc.*) of the user. The Aware Home[1], Place Lab[17], Smart Meeting Room[6], and vehicles[19] provide intelligent and adaptive service environment for assisting users to concentrate on their specific tasks.

Context-awareness is the essential characteristic of a smart space, and using the technologies to achieve context-awareness is a type of intelligent computing. Within a richly equipped and networked environment, users need not carry any devices with them; instead, applications adapt the available resources to their processes for delivering services to vicinity of users, as well as tracking the location of users. Cyberguide[18] uses the user's locations to provide an interactive map service. In the Active Badge[23], every user wears a small infrared device, which generates a unique signal and

can be used to identify the user. Xerox PARCTab[24] is a personal digital assistant that uses an infrared cellular network for communication. Bat Teleporting[15] is an ultrasound indoor location system.

In a smart space, augmented appliances, stationary computers, and mobile sensors can be used to capture raw contextual information (*e.g.* temperature, spatial data, network measurement, and environmental factor), and consequently a context-aware system needs to understand the meaning of a context. Therefore, a model to represent contextual information is the first issue of developing context-aware systems. Context-aware services require the high-level description about the user's states and environment situations. However, high-level context cannot be directly acquired from sensors. The capability to entail high-level contexts from the existing knowledge is required in context-aware systems. Consequently, how to derive high-level contexts is the second issue. As we know that people may move to anywhere at anytime, it is increasingly important that computers develop a sense of location and context in order to appropriately respond to the user's needs. How to deliver right services to right places at the right time will be the third issue. Inconsistent contexts may appear in context-aware systems due to systems should react to the rapid change of contextual information. Any systems with inconsistent knowledge will cause them fail to provide correct services. Therefore, a context-aware system must maintain a consistency knowledge base and react to the dynamic change of contexts, which will be the fourth issue.

In this research, we leverage multi-agent and semantic web technologies that provides the means to express context and uses abstract representations to derive usable context for proactively delivering context-aware service to the user. We propose an ontology-base model for supporting context management, which can provide high-level context reasoning and detect the knowledge inconsistency. In addition, a *Smart Alarm Clock* scenario is help for describing the detailed of our research.

2. BACKGROUND TECHNOLOGIES

An overview of the context models, context reasoning, and ontology are introduced in this section.

2.1 Context Representation

Context is mainly characterized by four dimensions[9]: location, identity, activity and time. Location refers to the

exact position where the user is. If we know a person's identity, we could easily derive related information from several data sources such as birth date, social connectivity, or email addresses. Knowing the location of an entity, we could determine its nearby objects and people.

Many context-aware systems concentrate on location aware services. Ye *et al.*[26] use lattice model to represent spatial structure, which can deal with syntactic and semantic labels. This general spatial model provides both absolute and relative references for geographic positions, both the containment and connection relationships can be determined as well. MINDSWAP Group at University of Maryland Institute for Advanced Computer Studies develops Semantic geoStuff¹ to express basic geographic features such as countries, cities, and relationships between these spatial descriptors.

The RFC 2445² defines iCalendar format for calendaring and scheduling applications, which provides users to create personal activities. Google Calendar³ is a popular web-based calendar supports iCalendar standard and users can share their own personal activities with others. These human activities are related to people, time, and location. Consequently, the contents of persons' schedules can help us to derive their location at a given time.

2.2 Ontology

Strang and Linnhoff-popien[21] concluded that the ontology are the most expressive model. Gruber[13] defines ontology as an "explicit specification of a conceptualization". Ontology is developed to capture the conceptual understanding of the domain in a generic way and provide a semantic basis for grounding the fine-grained knowledge.

COBRA-ONT[5] provides key requirements for modeling context in a smart meeting application. It defines concepts and relations of physical locations, time, people, software agents, mobile devices, and meeting events. SOUPA[7] (Standard Ontology for Ubiquitous and Pervasive Applications) uses some other standard domain ontologies, such as FOAF⁴ (Friend of A Friend), OpenGIS, spatial relations in OpenCyc, ISO 8601 date and time formats⁵, and DAML time ontology[16]. Clearly, these ontologies provide not only a rich context representation, but also make use of the abilities of reasoning and sharing knowledge.

2.3 Context reasoning

Design and implementation of context reasoning can vary depending on types of contextual information that are involved. Early context-aware systems[8, 25, 3] are tightly coded logics of context reasoning into the behavior of systems. Implementation for understanding the contextual information is bound into the programs. Therefore, developed applications often have rigid implementations and are difficult to maintain.

¹<http://www.mindswap.org/2004/geo/geoStuff.shtml>

²<http://tools.ietf.org/html/rfc2445>

³<http://calendar.google.com>

⁴<http://xmlns.com/foaf/spec/>

⁵<http://www.w3.org/TR/NOTE-datetime>

Rule-based logical inference can help to develop flexible context-aware systems by separating high-level context reasoning from low-level system behaviors. However, context modeling languages are used to represent contextual information and the rule languages are used to enable context reasoning. Accordingly, in most cases, these two types of languages have different syntax and semantic representations; it is a challenge that effectively integrates these distinctive languages to support context-aware systems. A mechanism to convert between contextual modeling and reasoning languages is one of solutions for this challenge. Gandon and Sadeh[11, 12] propose e-Wallet that implements ontologies as context repositories and uses a rule engine Jess[10] to invoke the corresponding access control rules. The e-Wallet using RDF⁶ triples to represent contextual information and OWL⁷ to define context ontology. Contextual information is loaded into the e-Wallet by using a set of XSLT⁸ stylesheets to translate OWL input files into Jess assertions and rules.

Ontology models can represent contextual information and specify concepts, subconcepts, relations, properties, and facts in a smart space. Moreover, ontologies reasoning can use these relations to infer the facts that are not explicitly stated in the knowledge base. Ranganathan *et al.*[20] propose that ontologies can make it easier to develop programs for reasoning about context. Chen[4] proposes that the OWL language can provide a uniformed solution for context representation and reasoning, knowledge sharing, and meta-language definitions. Anagnostopoulos *et al.*[2] adopt the Description Logics the most useful language for expressing and reasoning contextual knowledge. The OWL DL was designed to support the existing Description Logic business segment and has desirable computational properties for reasoning systems. Typical ontology-based context-aware application is EasyMeeting that uses OWL to define the SOUPA ontology and OWL DL to support context reasoning. Gu *et al.*[14, 22] propose an OWL encoded context ontology CONON in Service Orientated Context Aware Middleware (SOCAM). CONON consists two layers of ontologies, an upper ontology that focuses on capturing general concepts and a domain specific ontology. EasyMeeting and SOCAM are use an OWL DL reasoning engine to check the consistency of contextual information and provide further reasoning over low-level context to derive high-level context.

3. SYSTEM ARCHITECTURE

Figure 1 shows our Context-aware System Architecture, which can continuously proceeds changing contexts and proactively provides services to the user. The top part of Figure 1 depicts a smart space environment, which equipped with devices and applications, such as personal calendar, weather forecasts, location tracking system, contact list, and shopping list, as well as raw sensing data, can provide contextual information and deliver context-aware services. The Context Collection Agents obtain raw sensing data from the context sources and convert the raw context into a semantic representation. Each Context Collection Agent will deliver the sensed contextual information to the *Context Management* after receiving sensing data.

⁶<http://www.w3.org/TR/rdf-concepts/>

⁷<http://www.w3.org/TR/owl-features/>

⁸<http://www.w3.org/TR/xslt>

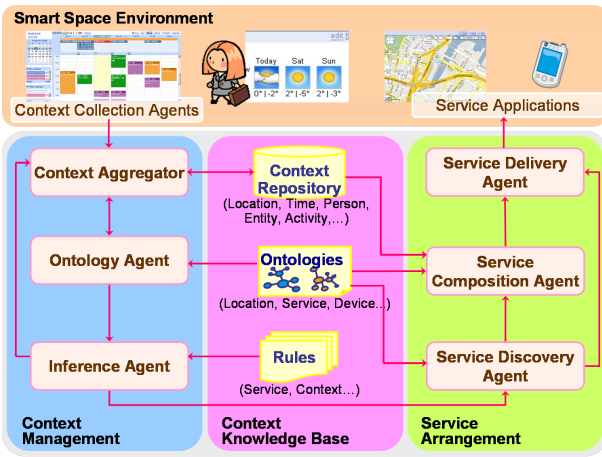


Figure 1: A Multi-agent System Architecture

The lower part of Figure 1 illustrates our context-aware system architecture, which consists of three components: *Context Management*, *Context Knowledge Base*, and *Service Arrangement*. Functions of *Context Management* are monitoring the contextual information and managing the environmental resources. Contextual information, domain knowledge, and service profiles are stored in *Context Knowledge Base*. *Service Arrangement* performs service discovery, composition, and execution. After assign the specified services, the system will invoke Service Applications to provide service in the smart space environment.

4. CONTEXT-AWARE AGENTS

Agents in Figure 1 accomplish the functions of managing contextual information and delivering context-aware services. Functions of *Context Management* include gathers contexts from the surrounding environment, provides methods for querying and storing the contextual information, and provides the context in an ontology-based representation that facilitates knowledge representation and inference. *Service Arrangement* checks whether any service operation can match the request under the current situation. If no operation matches the request, it combines operations to match the request.

4.1 Context Aggregator

Context Aggregator collects contextual information from Context Collection Agents and stores the context to the Context Repository for context inference, consistency checking, and knowledge sharing. There are two types of input context, the raw context and the high-level context. Raw context refers to the sensing data which directly obtained from context sources. For example, bed sensors can provide lay-on-bed sensing and weather forecast API can provide forecasting information. Context Aggregator subscribes to the specified Context Collection Agents for retrieving the contextual information, which define in the context ontology. Consequently, Context Collection Agents are the providers of low-level contexts while the high-level contexts are derived from the Ontology Agent and Inference Agent.

4.2 Ontology Agent

Ontology Agent loads and parses an OWL context ontology into RDF triples, which makes other agents able to represent and share context in the system. Context Aggregator sends the current state of contexts to Ontology Agent while the subscribed context changed. The other agents can send their queries to Ontology Agent for retrieving the updated knowledge. According to the structures and relationships between contexts that define in the context ontology, the Ontology Agent performs the subsumption reasoning for deducing new contextual information. For example, it can deduce the superclasses of a specified class and decides whether one class is subsumed by another, *e.g.*, a building may spatially subsume a room.

4.3 Inference Agent

Inference Agent adopts an OWL DL reasoning engine for supporting context reasoning and conflict detection. When Inference Agent receives contextual information from Ontology Agent, the reasoning engine will fire rules and trigger actions that may deduce new high-level contexts and derive service requests. Combining the inferred contexts with the original context ontology, Inference Agent can detect the context inconsistency. Either new high-level contexts or service requests can be derived from Inference Agent and deliver to Context Aggregator or Service Discovery Agent, respectively.

4.4 Service Discovery Agent

Service Discovery Agent maintains the service ontology. An OWL-S⁹ file defines the service ontology, which includes three essential types of knowledge about a service: service profile, process model, and service grounding. The 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 functionalities offer, which can be exploited to solve the goals. The role of service grounding is to provide concrete details of message formats and protocols. According to the description in service ontology, Service Discovery Agent keeps the atomic services information. When the Service Discovery Agent receives a service request, it checks whether any single service satisfies the requirement under the current situation. If an atomic service can accomplish the request, the associated service grounding information will be delivered to the Service Delivery Agent.

4.5 Service Composition Agent

If a service request cannot be achieved by a single service, Service Composition Agent will compose atomic services to fulfill the request. The service profile of a service ontology defines the service goals, preconditions, and effects. According to these semantic annotations, AI planning has been investigated for composing services. The state transition is defined by the operations, which consist of preconditions and effects. Initial states of the AI planner are combined the current contexts and context ontology. The service request is the planning goal. Therefore, giving initial states, goals, and operations, Service Composition Agent will derive a service execution plan, which is a sequence of operations that starts from initial states and accomplishes the given goal.

⁹<http://www.w3.org/Submission/OWL-S/>

4.6 Service Delivery agent

Service ontology defines the information for service grounding, which specifies the details of how an agent can access a service. According to the description of service grounding, Service Delivery Agent invokes the specified Service Application with the required protocol and message contents.

5. CONTEXT ONTOLOGY MODEL

Context-aware applications need a unified context model that is flexible, extendible, and expressive to adapt the variety of context features and dependency relations. The ontology models can fulfill these requirements; therefore, we deploy an ontology context model to represent contextual information in smart space environment. The ontology is inspired by the need to share knowledge about locations, time, and activities so that context-aware applications can infer the environmental contexts and trigger services.

5.1 Context Repository

Context Repository stores a set of consistent context, which including location, person, and activity information. Either raw or high-level context has a unique type identity and value. The associated value is the timestamp represents when the corresponding context is arrived. Context ontology defines the classes of contexts and the relationships between the instances of context objects. A RDF-triple represents a context that contains 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. For example, the following triple represents “Peter is sleeping”.

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

Where *Peter* represents subject, *participatesIn* is a predicate, and the activity *sleeping* is an object. We use subject and predicate as the compound key of Context Repository. When a context has been updated, the associated timestamp will be changed accordingly.

5.2 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 define a context ontology depicts in Figure 2 as a representation of common concepts about the smart space environment. Context information are collected from real-world classes (Person, Location, Sensor, Time, HomeEntity), and a conceptual class Activity. The class hierarchy represents an *is-a* relation; an arrow points from a subclass to another superclass. A class can have subclasses that represent the concepts more specific than their superclass. For example, we can divide the classes of all locations into indoor and outdoor locations, that is, Indoor Location and Outdoor Location are two disjoint classes and both of them belong to Location class. In addition, the subclass relation is transitive, therefore, the Livingroom is a subclass of Location class because Livingroom is a subclass of Indoor and Indoor is a subclass of Location.

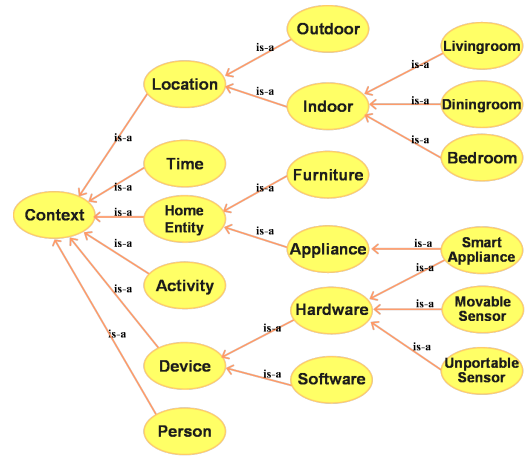


Figure 2: A Context Ontology

The relationship between classes is illustrated in Figure 3. The solid arrows describe relation between subject resources and object resources. For example, *isLocatedIn* describes the relation between the instances of Person and Location while the instances of Person is the subject resources and instances of Location is the object resources.

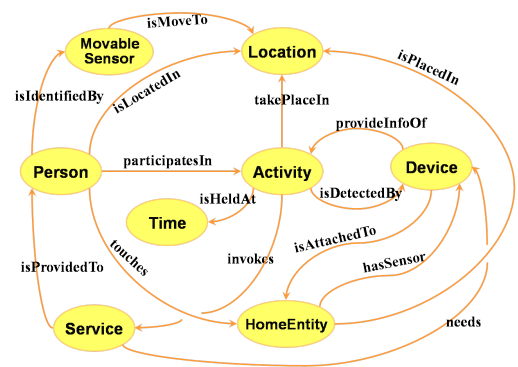


Figure 3: Context Relationship

A service ontology defined by OWL-S is for describing available services that comprises service profile, service model, and service grounding.

5.3 Rules

Rules of a rule-based system serve as IF-THEN statements. Context rules can be triggered to infer high-level context. According to the description of Figure 3, a rule for detecting the location of a user is showed as follows:

```
[Person_Location:
  (?person isIdentifiedBy ?tag)
  (?tag isMoveTo ?room)
->
  (?person isLocatedIn ?room)
]
```

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

Rule `Person_Location` is an example that can deduce high-level context. The `?person` is an instance of class `Person`, `?tag` is an instance of `MovableSensor`, and `?room` is an instance of `Room`, the rule `Person_Location` declares that if any person `?person` is identified by a movable sensor `?tag` and this movable sensor is move to a room `?room`, we can deduce that `?person` is located in `?room`.

6. CONTEXT MANAGEMENT AND REASONING MECHANISM

In order to make our research easier to understand, we use a simple example to describe the detail mechanism of context management and reasoning.

In a smart space, a Smart Alarm Clock can check Peter's schedule and automatically set the wake-up alarm for helping him not miss his 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 earlier then the alarm time, the alarm will be disabled.

6.1 Context Reasoning

In order to archive *Smart Alarm Clock*, we have to collect Peter's schedule to decide the alarm time and should reasoning whether Peter is awake or not. Google Calendar Data API¹⁰ can support the information of Peter's calendar events. The position-aware sensors, bed pressure sensors, *etc.* can help to detect whether user on the bed or not. For example, RFID technologies can be used to recognize and identify the activities of Peter. Using a wireless-based indoor location tracking system can determine Peter's location with room-level precision.

Figure 4 shows the instance relationships for detecting whether Peter is currently sleeping or not. The word within an oval represents a class and the box represents an instance of the corresponding class. For example, `bed` is an instance of `Furniture` class. Dashed line indicates the connection of a class and its instance. Each solid arrow reflects the direction of object property relationship that directs from domain to range. In addition, an inverse property can be declared while reverse the direction of a line. For example, the inverse object property of `isAttachedTo` is `hasSensor`.

A boolean data type property `isOn` is associated with `Sensor` class for detecting whether the value of instances is on or off. If someone is on the bed, value of the sensor `bed_sensor` will be on, that is, the value of `isOn` is `true`. Otherwise, when nobody touches the bed, the value of `isOn` has to be `false`. When an event of wake-up call has been triggered, a rule

¹⁰<http://code.google.com/apis/calendar/>

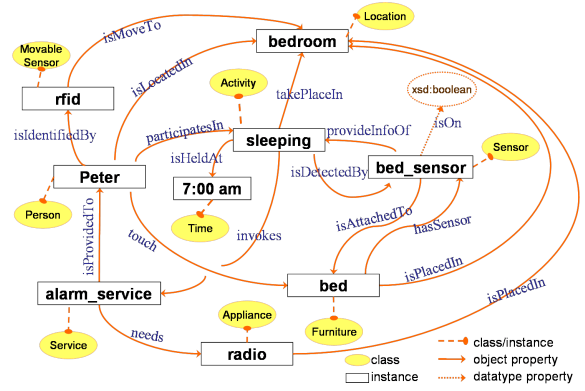


Figure 4: A Context Snapshot

for detecting the value of `bed_sensor` can be used to decide whether it is necessary to deliver the `alarm_service` or not.

For reasoning high-level contexts, we apply rule-based reasoning with horn clauses into the ontology model. The rule `Person_activity` can deduce what's the activity the user currently involved. For example, in Figure 4, when the time is up, given the location of Peter and the status of bed sensor, the rule `Person_activity` will be triggered and can deduce whether Peter is sleeping or not. The rule `Invoke_service` can deduce what's the service for delivering to the user. Given the instances of Figure 4, the rule `Invoke_service` reflects "if Peter is sleeping, deliver smart alarm service".

```
[Person_activity:
  (?person touch ?entity)
  (?entity hasSensor ?sensor)
  (?sensor providesInfoOf ?activity)
->
  (?person participatesIn ?activity) ]
```

```
[Invoke_service:
  (?person participatesIn ?activity)
  (?activity invokes ?service)
->
  (?service isProvidedTo ?person)]
```

6.2 Context Management

Changes of environmental contexts are transient in the sense of that any context may appear and vanish at anytime. Algorithm 1 shows how the Context Aggregator manages the contextual information.

We use RDF-triple to represent a context while an associated compound key comprises the subject and object. When a new context is arrived, Context Aggregator uses the key of new context to query Context Repository. If a context exists in Context Repository and the associated predicate represents one-to-one relationship, the new context will replace the old one. Otherwise, the new context will be inserted into Context Repository. Functions `update(keyc, c)` and `insert(keyc, c)` perform the context replacement and insertion, respectively. When a context is vanished, it should

Algorithm 1 Maintaining Context Repository

```
1: Input:  $c$  is the new context
2:  $C$ : Context Repository
3:  $rdfi$ : RDF-triple  $(s_i, p_i, o_i)$  of a context  $i$ 
4:  $key_i$ : key of context  $i$  in Context Repository
5: for all  $i \in C$  do
6:   if  $isOutdated(i)$  then
7:      $delete(i)$ 
8:   end if
9: end for
10: if  $\exists i \in C$  s.t.  $key_i = key_c$  and  $isOne2One(p_c)$  then
11:    $update(key_c, c)$ 
12: else
13:    $insert(key_c, c)$ 
14: end if
```

be removed. function $delete(i)$ can remove the specified context from Context Repository. We use a decay function to determine the existence of a context. Different context is associated with a different decay function. This function can either be an objective function for predicating a specified activity or simply be a constant function. The function $isOutdated(i)$ apply the context decay function to decide whether the context is existed or not.

6.3 Inconsistency Resolution

The Context Repository is dynamically updated for reflecting the change of context. Therefore, we must ensure incorrect or outdated contexts are not existed in Context Repository. If a raw context is changed, some of the inferred high-level contexts may be changed. For example, if Peter walks from living room to bedroom, the corresponding RDF-triple will be changed from $\langle \text{Peter isLocatedIn living_room} \rangle$ to $\langle \text{Peter isLocatedIn bedroom} \rangle$. Context Aggregator will update the location context of Peter because the property $isLocatedIn$ is one-to-one relationship. If a predicate allow multiple relationships, the original context will be reserved.

It is a challenge that when a raw context is changed, we need to updated the associated high-level contexts. However, it is hard to find the corresponding high-level contexts by using the context dependency of inference rules. Updating a context may easily trigger the infinite context dependency checking and can lead to unpredictable situations. We categorize the data in Context Repository to three types: core knowledge, raw-level context, and high-level context. The OWL ontologies define the contents of core knowledge that are static and persistent. The raw-level context is the raw sensing data that delivers from the Context Collection Agents in Figure 1. Using the core knowledge and raw-level context, a rule-based reasoning can deduce high-level contexts. When a raw context has been removed, we discard the original set of high-level context and perform context reasoning. Clearly, the deduced high-level contexts are consistent with the current raw contexts and the Context Repository can maintain the context consistency. This approach is simple, but can efficiently resolve the context inconsistency without recursively check the context dependency.

7. IMPLEMENTATION

Our agent is deployed on JADE¹¹ (Java Agent DEvelopment Framework), which is a FIPA-compliant software framework for multi-agent systems, implemented in Java and comprised several agents. Jena¹² is a Java framework for building Semantic Web applications, is used for providing a programmatic environment for RDF, RDFS, and OWL. Moreover, we use an open-source OWL Description Logics (OWL DL) reasoner Pellet¹³ that developed by Mindswap Lab at University of Maryland, to infer high-level contexts and detect context conflicts.

8. CONCLUSION AND FUTURE WORK

This research presents a context management mechanism in a smart space. We integrate context-aware technologies, semantic web, and logical reasoning for providing context-aware services. An ontology-based model supports reasoning mechanism, which can deduce high-level contexts and detect context consistency.

We use a simple scenario to demonstrate the mechanism of context management. However, this simple case does not show the power of context reasoning. Therefore, design other scenarios that can explain and evaluate our approach is one of our future direction.

9. 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] C. B. Anagnostopoulos, A. Tsounis, and S. Hadjiefthymiades. Context awareness in mobile computing environments. *Wireless Personal Communications: An International Journal*, 42(3):445–464, 2007.
- [3] L. Capra, W. Emmerich, and C. Mascolo. CARISMA: Context-aware reflective middleware system for mobile applications. *IEEE Transactions on Software Engineering*, 29(10):929 – 945, 2003.
- [4] H. Chen. *An Intelligent Broker Architecture for Pervasive Context-Aware Systems*. PhD thesis, University of Maryland, Baltimore County, 2004.
- [5] H. Chen, T. Finin, and A. Joshi. An ontology for context-aware pervasive computing environments. *The Knowledge Engineering Review*, 18(3):197–207, September 2003.
- [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] H. Chen, F. Perich, T. Finin, and A. Joshi. SOUPA: Standard ontology for ubiquitous and pervasive applications. In *The First Annual International Conference on Mobile and Ubiquitous Systems:*

¹¹<http://jade.tilab.com/>

¹²<http://jena.sourceforge.net/>

¹³<http://pellet.owldl.com/>

- Networking and Services (MobiQuitous'04)*, pages 258–267, August 2004.
- [8] M. H. Coen. Building brains for rooms: designing distributed software agents. In *Proceedings of the Conference on Innovative Applications of Artificial Intelligence (IAAI'97)*, pages 971–977. AAAI Press, 1997.
- [9] A. K. Dey. *Providing architectural support for building context-aware applications*. PhD thesis, Georgia Institute of Technology, 2000. Director-Gregory D. Abowd.
- [10] E. Friedman-Hill. *Jess in Action: Java Rule-Based Systems*. Manning Publications, Greenwich, CT, USA, 2003.
- [11] F. L. Gandon and N. M. Sadeh. A semantic e-wallet to reconcile privacy and context awareness. *Lecture Notes in Computer Science: The Semantic Web (ISWC 2003)*, 2870:385–401, October 2003.
- [12] F. L. Gandon and N. M. Sadeh. Semantic web technologies to reconcile privacy and context awareness. *Journal of Web Semantics*, 1(3):241–260, 2004.
- [13] T. R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, June 1993. Special issue: Current issues in knowledge modeling.
- [14] T. Gu, X. H. Wang, H. K. Pung, and D. Q. Zhang. An ontology-based context model in intelligent environments. In *Proceedings of Communication Networks and Distributed Systems Modeling and Simulation Conference*, pages 270–275, 2004.
- [15] A. Harter, A. Hopper, P. Steggle, A. Ward, and P. Webster. The anatomy of a context-aware application. *Wireless Networks*, 8(2–3):187–197, March – May 2002.
- [16] J. R. Hobbs and F. Pan. An ontology of time for the semantic web. *ACM Transactions on Asian Language Information Processing (TALIP)*, 3(1):66–85, 2004. Special Issue on Temporal Information Processing.
- [17] S. S. Intille. Designing a home of the future. *IEEE Pervasive Computing*, 1(2):76–82, April 2002.
- [18] S. Long, D. Aust, G. Abowd, and C. Atkeson. Cyberguide: prototyping context-aware mobile applications. In *Conference companion on Human factors in computing systems (CHI '96)*, pages 293 – 294, Vancouver, British Columbia, Canada, April 13 – 18 1996. ACM Press.
- [19] G. Look and H. Shrobe. A plan-based mission control center for autonomous vehicles. In *IUI '04: Proceedings of the 9th international conference on Intelligent user interfaces*, pages 277–279, New York, NY, USA, 2004. ACM Press.
- [20] A. Ranganathan, J. Al-Muhtadi, and R. H. Campbell. Reasoning about uncertain contexts in pervasive computing environments. *IEEE Pervasive Computing*, 3(2):62–70, 2004.
- [21] T. Strang and C. Linnhoff-popien. A context modeling survey. In *Workshop on Advanced Context Modelling, Reasoning and Management at The Sixth International Conference on Ubiquitous Computing (UbiComp 2004)*, Nottingham, England, 2004.
- [22] X. H. Wang, D. Q. Zhang, T. Gu, and H. K. Pung. Ontology based context modeling and reasoning using OWL. In *Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops (PERCOMW '04)*, page 18, Washington, DC, USA, 2004. IEEE Computer Society.
- [23] R. Want, A. Hopper, V. Falcão, and J. Gibbons. The active badge location system. *ACM Transactions on Information Systems (TOIS)*, 10(1):91–102, January 1992.
- [24] R. Want, B. N. Schilit, N. I. Adams, R. Gold, K. Petersen, D. Goldberg, J. R. Ellis, and M. Weiser. An overview of the PARCTAB ubiquitous computing experiment. *Personal Communications*, 2(6):28 – 43, December 1995.
- [25] H. Wu, M. Siegel, and S. Ablay. Sensor fusion for context understanding. In *Proceedings of IEEE Instrumentation and Measurement Technology Conference*, Anchorage, AK, USA, May 21 – 23 2002.
- [26] J. Ye, L. Coyle, S. Dobson, and P. Nixon. A unified semantics space model. In J. Hightower, B. Schiele, and T. Strang, editors, *Proceedings of the 3rd International Symposium on location- and Context-Awareness (LoCA 2007)*, volume 4718 of *Lecture Notes in Computer Science*, pages 103–120, September 2007.