# THE DESIGN AND IMPLEMENTATION OF SHARED ONTOLOGIES FOR SMART SPACE APPLICATION

JUN-FENG MAN[1,2], QING CHEN[1,2], XIAO-HENG DENG[1,2], YIN-AN QIU[1]

[1]Computer Science Department, Zhuzhou Institute of Technology, Zhuzhou 412008, China
[2]College of Information Science and Technology, Central South University, Changsha 410083, China
E-MAIL: mjfok@tom.com

**Abstract:**

**In smart space applications, we usually adopt semantic Web technologies for supporting pervasive context-aware ability to process many onerous tasks, e.g., knowledge sharing, context reasoning and interoperability. In order to achieve above purpose, it is necessary to exploit a Shared Ontologies for Pervasive Computing (SO4PC). These ontologies are expressed with standard Web Ontology Language(OWL) and include modular component vocabularies to represent intelligent agents with associated beliefs, desires, intentions, time, space, events, user profiles, actions, policies for security and private protection. We discuss how SO4PC can be extended and used to support the applications of Smart Meeting Rooms (SMR) which is a broker-centric agent architecture.**

**Keywords:**

**Semantic Web; Ontology; Pervasive computing; Context; Reasoning**

## 1. Introduction

Pervasive computing is a natural extension of the existing computing paradigm. In the pervasive computing vision, computer systems will seamlessly integrate into the everyday life of users and provide them with services and information in an "anywhere, anytime" fashion. In this open and dynamic environment, intelligent computing entities must be able to share knowledge, reason about their environment, and interoperate.

In the past, many prototyping systems and architectures have been successfully developed, e.g., handheld devices are augmented with context-aware applications to create personalized tour guides for the museum visitors[1], the user interfaces and applications on a resource-poor mobile device can dynamically migrate to a nearby resource-rich stationary computer when the user enters the office[2]. However, they offer only weak support for knowledge sharing and reasoning.

Main weakness of the above systems and architectures is that they are not built on the foundation of common ontologies with explicitly semantic representation[3]. For example, the location information of a user is widely used for guiding the adaptive behavior of the systems[4]. However, none has taken advantageous of the semantics of spatial relations in reasoning about the location context of users. Additionally, many systems use programming language objects (e.g., Java class objects) to represent the knowledge that the computer systems have about the situational environment. Because these representations require the establishment of a prior low-level implementation agreement between the programs that wish to share information, they cannot facilitate knowledge sharing in open and dynamic environment.

To address these issues, we believe a shared ontologies must be developed for supporting knowledge sharing, context reasoning and interoperability in pervasive computing systems. By defining such ontologies, we can help system developers to reduce their efforts in creating ontologies and to be more focused on the actual system implementations.

## 2. Background

In previous systems, ontologies are often defined based on ad hoc representation schemes such as a set of programming language objects or data structures. There are two problems in this approach: (1) the use of ad hoc representation schemes lacking shared vocabularies hinders the ability of independently developed agents to interoperate (i.e., to share context knowledge), and (2) the use of objects and data structures of low expressive power provides inadequate support for context reasoning.

In order to help agents to discover, reason about and communicate contextual information, we must define explicit ontologies for context concepts and knowledge. In this paper, we present a set of ontologies that we have developed to support ubiquitous agents in the Context

0-7803-9091-1/05/$20.00 ©2005 IEEE

Broker Architecture(CBA) system. CBA is a broker-centric agent architecture for smart space applications. Being suit for CBA, we exploit a set of ontologies which is defined in the Ontology Web Language (OWL)[5]. The current version of the CBA ontology models the basic concepts of people, agents, places and presentation events. It also describes the properties and relationships between these basic concepts including (1) relationships between places, (2) roles associated with people in presentation events, and (3) typical intentions and desires of speakers and audiences.

In figure 1, The center of CBA is an intelligent agent called the context broker. In a smart space, a context broker has the following responsibilities:

(1) Provide a centralized model of the context that all devices, services, and agents in the space can share,

(2) Acquire contextual information from sources that are unreachable for the resource-limited devices,

(3) Reason about contextual information that can't be directly acquired from the sensors,

(4) Detect and resolve inconsistent knowledge stored in the shared context model,

(5) Protect privacy by enforcing policies that users have defined to control sharing and using of their contextual information.

In following sections, we will discuss how to create a set of ontologies expressed with OWL to support context modeling and knowledge sharing, detect and resolve inconsistent context knowledge.

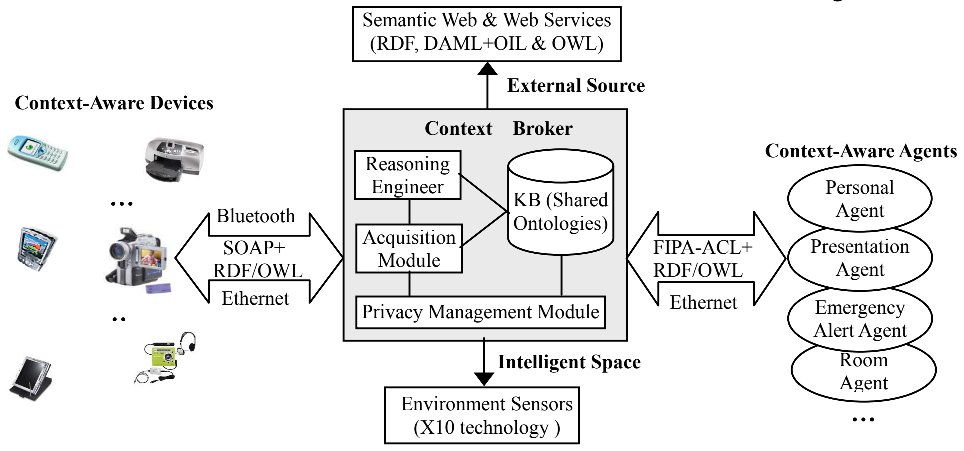


Figure 1. The architecture of context broker

## 3. SO4PC Ontologies

### 3.1. The Ontology Web Language - OWL

The OWL language is a Semantic Web language used by computer applications that need to process the content of information instead of just presenting information to humans[6]. This language is developed in part of the Semantic Web initiatives sponsored by the World Wide Web Consortium (W3C).

OWL is a knowledge representation language for defining and instantiating ontologies. An ontology is a formally explicit description of concepts in a domain of discourse (or classes), properties of each class describing various features and attributes of the class, and restrictions on properties[7].

The normative OWL exchange syntax is RDF/XML. Ontologies expressed with OWL are usually placed on web servers as web documents, which can be referenced by other ontologies and downloaded by applications that use ontologies.

### 3.2. Related Ontologies

Part of the SO4PC vocabularies are adopted from a number of different consensus ontologies. The strategy for developing SO4PC is to borrow terms from these ontologies but not to import them directly. Although the semantics for importing ontologies is well defined, by choosing not to use this approach we can effectively limit the overhead in requiring reasoning engines to import ontologies that may be irrelevant to pervasive computing applications. However, in order to allow better interoperability between the SO4PC applications and other ontology applications, many borrowed terms in SO4PC are mapped to the foreign ontology terms using the standard OWL ontology mapping constructs (e.g., owl:equivalentClass and owl:equivalentProperty).

The ontologies that are referenced by SO4PC include the Friend-Of-A-Friend ontology (FOAF)[8], DAML-Time and the entry sub-ontology of time[9], the spatial ontologies in OpenCyc[10], Regional Connection Calculus (RCC)[11], COBRA-ONT[12], MoGATU BDI ontology[13], and the Rei policy ontology[14]. In the rest of this section, we will describe the key features of these related ontologies and point out their relevance to pervasive computing applications.

*FOAF* This ontology allows the expression of personal information and relationships, and is a useful building block for creating information systems that support online communities. Pervasive computing applications can use FOAF ontology to express and reason about a person's contact profile and social connections to other people in their close vicinity.

*DAML-Time & the Entry Sub-ontology of Time* The vocabularies of these ontologies are designed for expressing temporal concepts and properties common to any formalization of time. Pervasive computing applications can use these ontologies to share a common representation of time and to reason about the temporal orders of different events.

*OpenCyc Spatial Ontologies & RCC* The OpenCyc spatial ontologies define a comprehensive set of vocabularies for symbolic representation of space. The ontology of RCC consists of vocabularies to express spatial relations for qualitative spatial reasoning. In pervasive computing applications, these ontologies can be exploited for describing and reasoning about location and location context.

*COBRA-ONT & MoGATU BDI Ontology* Both COBRA-ONT ontology and MoGATU BDI ontology are aimed for supporting knowledge representation and ontology reasoning in pervasive computing environment. While the design of COBRA-ONT focuses on modeling contexts in smart meeting rooms (SMR), the design of MoGATU BDI ontology focuses on modeling the belief, desire, and intention of human users and software agents.

*Rei Policy Ontology* The Rei policy language defines a set of deontic concepts (i.e., rights, prohibitions, obligations, and dispensations) to specify and reason about security access control rules. In pervasive computing environment, users can use this policy ontology to specify high level rules for granting and revoking the access rights to and from different services.

SO4PC consists of two parts: SO4PC Core and SO4PC Extension. The set of the SO4PC Core ontologies defines generic vocabularies that are universal for different pervasive computing applications. The set of SO4PC Extension ontologies extended from the core ontologies defines additional vocabularies for supporting specific types of applications and provides examples for the future ontology extension.

## 3.3. SO4PC

This set of ontologies consists of vocabularies for expressing concepts that are associated with person, agent, belief-desire-intention (BDI), action, policy, time, space, and event. The ontologies are grouped into nine distinctive ontology documents. Figure 2 shows a diagram of the ontology documents and their associated relations.

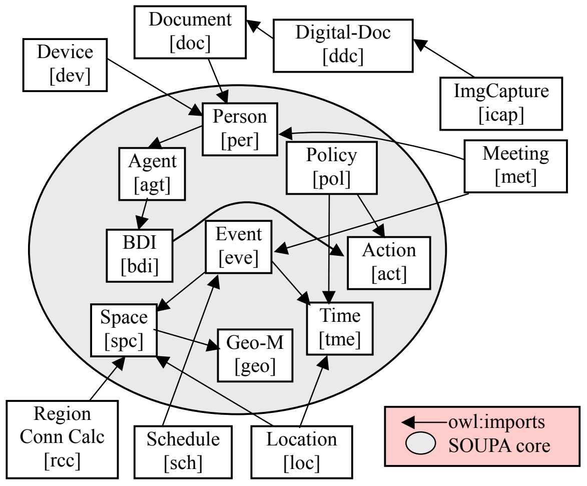


Figure 2. The relation of SO4PC

Here, we will simply discuss SO4PC by some samples of SMR.

*Person.* The "Person" class defines the most general properties about a person in a smart space. In CBA, each person is required to have a name, an email address and a homepage URL. The subclasses of the "Person" class include:

(1) The "PersonInBuilding" class which defines a type of people who are currently in a building,

(2) The "MeetingParticipant" class which defines a type of people who are currently attending a meeting,

(3) The "TalkEventHost" class which defines a type of people who have invited speakers to a meeting,

(4) The "Speaker" class which defines a type of people who are invited by other people to give a speech or presentation at a meeting

(5) The "Audience" class which defines a type of people who are meeting participants, but not speakers.

*Place.* The "Place" class defines the containment relationship properties (i.e., isPartOf and hasPartOf) and naming properties of a place. The naming properties require all “Place” instances to be associated with a full address name and an alias address name for shorthand. The containment relationship defines that an instance of the “Place” class may contain any number of instances of the “Place” class, and it may also be contained by other instances of the “Place” class.

There are four subclasses of the “Place” class in this version of the ontology. They are “Campus”, “Building”, “Room”, “OtherPlaceInBuilding” and “MeetingPlaceInBuilding”. Because each of these subclasses has specialized features, in their class definitions, they further restrict the inherited containment properties as the following:

(1) The “Campus” class represents the notion of a school or campus. This class restricts the cardinality of the “isPartOf” property to 0, which means a campus is a type of “Place” that is not contained by any other place. On the other hand, this class restricts the range value of the “hasPartOf” property to be a type of “Building”, which means only instances of the type of “Building” can be contained by an instance of this class.

(2) The “Building” class represents the notion of buildings on a university campus. This class restricts the maximum cardinality of the “isPartOf” property to 1 and the range value to be the class type “Campus”, which means a building can only be a part of at most one “Campus” instance. On the other hand, this class restricts the “hasPartOf” property to have minimum cardinality to 1 and the range value to be the class type “Room”, which means a building must contain one or more “Room” instances.

(3) The “Room” class represents the notion of rooms in a building. This class restricts the maximum cardinality of the “isPartOf” property to 1 and the range value to be the class type “Building”, which means a room can only be a part of one building at most. On the other hand, this class restricts the cardinality of the “hasPartOf” property to 0, which means the room is a special type of “Place” that cann’t contain any instance of the “Place” class or its subclasses.

(4) The “OtherPlaceInBuilding” class represents places in a building that are not usually categorized as a type of room (e.g., hallways and cafeteria). In addition, being a subclass of the “Place” class, this class is defined as a class which disjoints from the “Room” class (using OWL’s disjoint property). The cardinality of the containment properties of this class is the same as the “Room” class.

(5) The “MeetingPlaceInBuilding” class represents rooms in which meeting events are currently taking place. The cardinality of the containment properties of this class is the same as the “Room” class. In addition to these properties, this class restricts an additional property “hostsMeeting”. This restriction provides a means to distinguish an instance being a specialized type of the “MeetingPlaceInBuilding” class from a generalized type of “Room” class.

*Action & Policy.* The “Action” and “Policy” classes define the notion of user intentions and corresponding policies, for example, a speaker’s intention to give presentation and an audience intention’s to receive a copy of presentation slides and handouts by registering the meeting.

*Time.* The “Time” class defines a set of properties for expressing time and temporal relations. They can be used to describe the temporal properties of different events that occur in physical world.

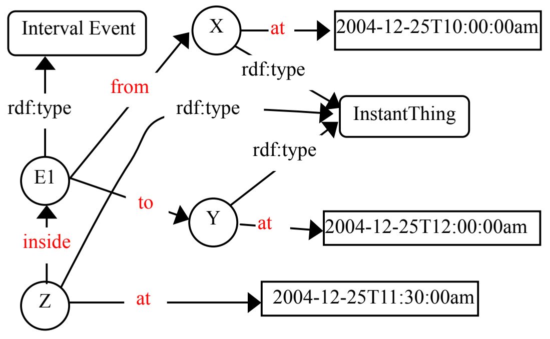


Figure 3. The sample of time instants and intervals

The basic representation of time consists of the tme:Time-Instant and tme:TimeInterval classes. The tme:Time-Instant is type of time instant, and tme:Interval-Thing is type of time interval. The union of these two classes forms the tme:TemporalThing class. In order to associate temporal things with date/time values(i.e., their temporal descriptions), the tme:at property is defined to associate an instance of the tme:Instant-Thing with an XML xsd:dateTime datatype value(e.g., 2004-12-25T14:30:00), and the tme:from and tme:to properties are defined to associate an instance of the IntervalThing with two different tme:Time-Instant individuals

## 4. Examples and use case

The key feature of SO4PC is the ability to support ontology reasoning in a distributed dynamic environment. The design of the ontology makes an open-world assumption about how and where distributed information is described. This means that properties about a particular person, place, and activity can be described by distributed heterogeneous sources, and the contexts of these individual entities can be dynamically inferred through classification.

In this section, we will describe three use cases that demonstrate how SO4PC can be used to support ontology reasoning in building a Context Broker for a smart meeting environment.

## 4.1. People Presence Reasoning

In a pervasive computing environment, sensors are often used to detect the presence of people in a building. For example, Bluetooth sensors can detect the proximity presence of the Bluetooth-enabled personal devices and conclude the presence of the device owners.

Using the SO4PC, these people presence sensors can effectively share people's presence information with the broker in the system and enable the broker to reason about the situational contexts of these people. For example, (1) Whether a person is in the building, (2) Whether a person is in school today, and (3) Whether a person is not in a room (e.g., in hallway or in a cafeteria).

Assuming the scenario, When Tom enters Room R101 and swipes his RFID badge at the door, the RFID sensor informs the broker of his presence in the room. On receiving this information, the broker will reason about Tom's context. The following three examples describe how the broker may reason about his contexts.

*Example 1*: To determine if Tom is in the Computer Building.

A1: Person("Tom") has property isCurrentlyIn("R101").

A2: For any person who has the property isCurrentlyIn() with rdfs:range limited to any Place that isPartOf Building, that person must be a type of PersonInBuilding (i.e., that person is in a building).

A3 <= A1+A2: Person("Tom") is a type of the PersonInBuilding class (i.e., Tom is currently in a building). Because Room("R101") isPartOf the Building("Computer Building"), the broker can deduce Tom is currently in the Computer building.

*Example 2*: To determine if Tom is in school today.

B1: Person("Tom") is in Building("Computer Building").

B2: Building("Computer Building") isPartOf Campus("Zhuzit").

B3 <= B1+B2: Person("Tom") is in school today.

*Example 3*: To determine if Tom is NOT in room R101.

C1: Person("Tom") is in Room("R201")

C2: If a person has property isCurrentlyIn with value that is a type of OtherPlaceInBuilding, then that person is not currently in the room. The class OtherPlaceInBuilding rdfs:disjointWith the class Room.

C3 <= C1+C2: Person("Tom") is NOT in room R101.

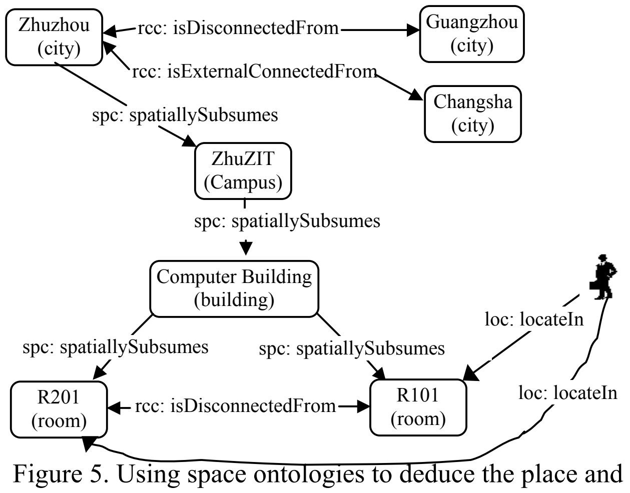
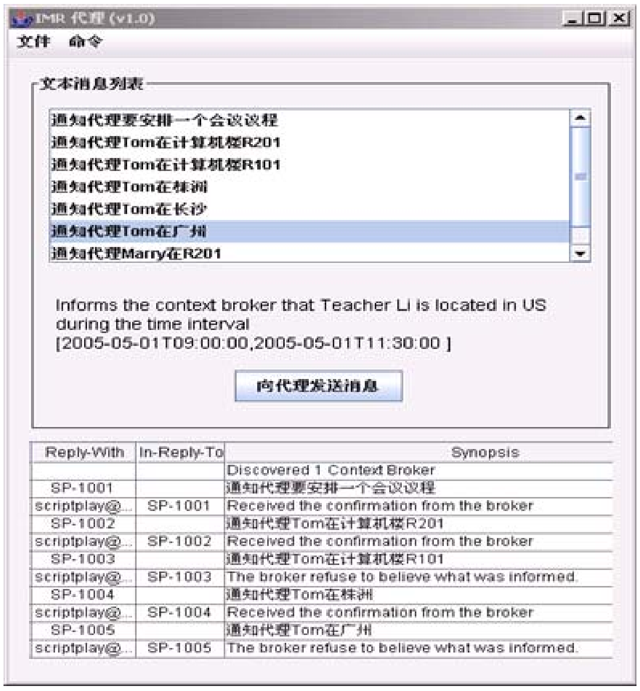


Figure 5. Using space ontologies to deduce the place and detect inconsistent location information

We use JADE's agent development toolkit to design a demonstration of presence reasoning.



通知代理要安排一个会议议程
通知代理Tom在计算机楼R201
通知代理Tom在计算机楼R101
通知代理Tom在株洲
通知代理Tom在长沙
通知代理Tom在广州
通知代理Marry在R201

Informs the context broker that Teacher Li is located in US during the time interval [2005-05-01T09:00:00,2005-05-01T11:30:00 ]

| Reply-With | In-Reply-To | Synopsis |
|---|---|---|
| | | Discovered 1 Context Broker |
| SP-1001 | | 通知代理要安排一个会议议程 |
| scriptplay@... | SP-1001 | Received the confirmation from the broker |
| SP-1002 | | 通知代理Tom在计算机楼R201 |
| scriptplay@... | SP-1002 | Received the confirmation from the broker |
| SP-1003 | | 通知代理Tom在计算机楼R101 |
| scriptplay@... | SP-1003 | The broker refuse to believe what was informed. |
| SP-1004 | | 通知代理Tom在株洲 |
| scriptplay@... | SP-1004 | Received the confirmation from the broker |
| SP-1005 | | 通知代理Tom在广州 |
| scriptplay@... | SP-1005 | The broker refuse to believe what was informed. |

Figure 6. The interface of presence reasoning program

## 4.2. A Room Agent

In SMR, room agents will play an important role in maintaining and sharing room-specific contexts with devices and agents. Let's assume in each room, there is a room agent maintaining a set of specific contexts of the room, for example (1) Whether the room is currently hosting a meeting, (2) The temperature, noise level, and light intensity level in the room, (3) The close/open states of the doors and windows in the room, (4) The type of devices/services that are available as the context of the room changes, the room agent will inform the broker of the updated contexts.

*Example 4*: To determine if Tom is currently in a meeting place in the Computer building.

D1: Person("Tom") is in Room("R201") and Building("Computer Building")

D2: For any room that has the property hostsMeeting() with rdfs:range limited to Meeting, the room must be a type of MeetingPlaceInBuilding.

D3: Room("R201") has the property hostMeeting("me239").

D4 <= D2+D3: Room("R201") is a type of Meeting-PlaceInBuilding.

D5: If a person is currently in a room, and that room is a type of MeetingPlaceInBuilding, then that person is currently in a meeting place.

D6 <= D1+D4+D5: Person("Tom") is currently in a meeting place which is in the Computer building.

*Example 5*: To determine if Tom is attending a meeting in room R201 (i.e., is Tom a meeting participant?).

E1: Person("Tom") is in Room("R201")

E2: Room("R201") is a type of MeetingPlaceIn-Building.

E3: If a person has the property isCurrentlyIn() with a value that is a type of Room class, then that person is a type of MeetingParticipant.

E4 <= E1+E2+E3: Person("Tom") is a meeting participant.

## 4.3. A Person Agent

Person agents are specialized agents that provide personalized services for individual users[15]. In smart space, these agents will keep track of users' profiles, preferences, desires and intentions. For example, the person agent of a speaker will automatically set up presentation slides when the speaker arrives at the meeting and adjust room lighting when the presentation starts. In order to provide these services to the person agent, it must acquire contextual knowledge about the person from the broker. This knowledge may include the following: (1) The role of the person in the meeting, (2) The type of services that the person has access to, (3) The type of the devices that the person carries, (4) The type of non-computing objects the person's vicinity (e.g., the type of clothes the person wears & the type of objects that the person holds), (5) The time at which the person enters the room or joins the meeting, (6) The identity of people whom the person is talking to.

One source from which person agent can acquire information about their users is through user behavior monitor. For example, Tom is scheduled to talk about ontology development at Wednesday's meeting. Days before the meeting, while Tom prepares his PowerPoint slides, his person agent learns his intention to give presentation at the meeting. On the day of the meeting, as Tom enters the conference room, the person agent informs the broker of Tom's intention and queries the broker for Tom's situational context. On receiving information from a person agent, the broker will reason about the context of the user. Sometimes ontology reasoning may involve uncertainty. For example, knowledge about the context of a person may not always be completely accurate. The following examples show how reasoning about the role of a person can involve varied degree of certainty.

*Example 6*: To determine the role of a person (e.g., is Tom is the speaker of meeting "me239").

F1: Person("Tom") is the same person as MeetingParticipant("Tom")

F2: MeetingParticipant("Tom") is associated with Meeting("me239")

F3: Person("Tom") has the intention to GiveSlideShowPresentation(Informed by Tom's person agent).

F4: If a person is a type of MeetingParticipant and that person has the SpeakerIntention, then that person is LIKELY to be a speaker.

F5 <= F1+F2+F3: Person("Tom") is likely to be a speaker.

Now, let's assume the broker has some prior knowledge about the invitations that are given to meeting participants. For example, from a talk announcement server (e.g., ITTalks.ORG[16]), the broker learns that some person who is a type of TalkEventHost has invited Tom to the meeting "me239".

F6: Person("Tom") is invited by a TalkEventHost.

F7 <= F5+F6: Person("Tom") must be a speaker.

## 4.4. Temporal Reasoning

Temporal ordering information is useful to a context broker in a number of different ways: (1) It can help a context broker to answer queries with temporal constraints. For example, an agent may wish to know the arrival time of different people who are present in a particular room from 13:00 to 15:00. (2) Temporal ordering information can help a context broker to correlate distinctive contextual information. For example, knowing a meeting is scheduled to take place from 13:00 to 15:00 in Room R101, and a cellphone device has been detected in the same room at 14:30, without any evidence to the contrary, a context broker can conclude the owner of the cellphone is attending the meeting. (3) Temporal ordering information can be also used to detect inconsistent contextual knowledge. For example, a context broker is informed that a person is currently located at home. Later, a new report indicates that the same person is present at work. The event time intervals described in the two reports overlap. Using this temporal ordering information, the context broker can detect the person's location information is inconsistent.

*Example 7*: To deduce owner of the cellphone is attending the meeting.

G1: In TimeInterval(13:00, 15:00), there is holding a meeting at Room("R201").

G2: It is 14:30(i.e., Time-Instant(14:30)).

G3: Context broker has detected DeviceAt("Tom", "cellphone", "R201").

G4: There is no evidence to confirm that Tom is at other place.

G5 <=G1+G2: G2 is a type of time instant of G1.

G6 <=G3+G4+G5: Tom is attending the meeting in Room R201 at 14:30.

## 5. Conclusions

In this paper, we have described a broker-centric architecture for an agent based on pervasive computing environment. This broker accepts and integrates context related information from devices and agents in the environment as well as from other sources to maintain a coherent model of the space, the devices, agents and people in it, and their associated services and activities. A key to realizing this architecture is the use of a set of common ontologies that the devices, agents and people use to describe their context information and query the broker's context model. For this reason, we exploit a set of ontologies – SO4PC,which is helpful for concepts description, context reasoning and knowledge share. In future, we plan to prototype an ontology reasoning component for building a Context Broker Architecture.

## References

[1] T. Kindberg, J. Barton. A web-based nomadic computing system. Computer Networks, 35(4):443–456, 2001.

[2] F. Bennett, T. Richardson, A. Harter. Teleporting – making applications mobile. In Proceedings of 1994 Workshop on Mobile Computing

[3] Man Junfeng, Liu Qiang, Yang Ding. Research on the Representation Method of Spatial Data[J]. Computer Application, 28(11):97-100, 2004

[4] N. B. Priyantha, A. Chakraborty, H. Balakrishnan. The cricket location-support system. In Mobile Computing and Networking, pages 32–43, 2000.

[5] F. van Harmelen, J. Hendler, I. Horrocks. Owl web ontology language reference. http://www.w3.org/TR/owl-ref/), 2002.

[6] D. L. McGuinness, F. van Harmelen. OWL Web Ontology Language Overview. In Proposed Recommendation (PR) for OWL. Web Ontology Working Group, W3C, 2003.

[7] N. F. Noy, D. L. McGuinness. Ontology development 101: A guide to creating your first ontology. Technical Report KSL-01-05, Stanford Knowledge Systems Laboratory, 2001.

[8] D. Brickley, L. Miller. FOAF vocabulary specification. In RDF Web Namespace Document, 2003.

[9] F. Pan, J. R. Hobbs. Time in OWL-S. In Proceedings of AAAI-04 Spring Symposium on SemanticWeb Services, Stanford University, California, 2004.

[10] D. B. Lenat, R. V. Guha. Building Large Knowledge-Based Systems: Representation and Inference in the Cyc Project. Addison-Wesley, February 1990.

[11] D. A. Randell, Z. Cui, A. Cohn. A spatial logic based on regions and connection. Morgan Kaufmann, San Mateo, California, pages 165–176, 1992.

[12] H. Chen, T. Finin, A. Joshi. An ontology for context-aware pervasive computing environments. Special Issue on Ontologies for Distributed Systems, Knowledge Engineering Review, 2003.

[13] F. Perich. MoGATU BDI Ontology, 2004.

[14] L. Kagal, T. Finin, A. Joshi. A Policy Based Approach to Security for the Semantic Web. In 2nd International Semantic Web Conference (ISWC2003), September 2003.

[15] T. Finin, A. Joshi, L. Kagal. Information Agents for Mobile and Embedded Devices. International Journal on Cooperative Information Systems, pages 30-35, 2003.

[16] S. Cost, T. Finin, A. Joshi, et.al., ITTALKS: A Case Study in the Semantic Web and DAML+OIL. IEEE Intelligent Systems, pages 40-47, 2002.