The Role of the Semantic Web in Pervasive Context-Aware Systems*

Harry Chen, Tim Finin, and Anupam Joshi

Department of Computer Science & Electrical Engineering
University of Maryland Baltimore County, Baltimore MD 21250, USA
{hchen4,finin,joshi}@cs.umbc.edu

Abstract. This paper describes an ontology expressed in OWL for supporting agent based context-aware systems in a pervasive computing environment. Defining an explicit representation of the ontology can help agents to reason about contexts and share context knowledge in a dynamic environment. Central to our architecture is the presence of an intelligent context broker that accepts context related information from devices and agents in the environment as well as from other sources, including information available on the Semantic Web describing the space and the activities scheduled to take place in it. The current version of the ontology models the basic concepts of people, agents, places and presentation events. Unlike the designs of Cyc and SUO, our ontology design is aimed to create a set of ontological vocabularies with basic semantics that are pragmatic for building pervasive context-aware systems in the near future. We present use cases of our ontology in an intelligent meeting room system and discuss associated reasoning mechanisms using Flora-2 in XSB.

1 Introduction

Computing is moving toward pervasive, ubiquitous environments in which devices, software agents, and services are all expected to seamlessly integrate and cooperate in support of human objectives – anticipating needs, negotiating for service, acting on our behalf, and delivering services in an anywhere, any-time fashion [26]. An important next step for pervasive computing is the integration of intelligent agents that employing knowledge and reasoning to understand the local context and share this information in support of intelligent applications and interfaces. We are developing a new pervasive context-aware computing infrastructure called Context Broker Architecture (CoBrA) [5], to support ubiquitous agents, services and devices to behave intelligently in according to their situational contexts.

A key requirement for realizing this infrastructure is the use of Semantic Web technology, which includes languages for building ontologies and tools for processing and reasoning over information described using these ontologies. These

* This paper is a draft submitted to ISWC03. This work was partially supported by DARPA contract F30602-97-1-0215 and Hewlett Packard.

emerging technologies are important to the future pervasive context-aware systems for the following reasons: (i) Semantic Web technologies can help devices and agents that were not designed to work together to interoperate, achieving "serendipitous interoperability" [11], (ii) ontologies provides a means for distributed agents to share context knowledge and to reason about contexts [6, 5], (iii) ontology languages can used to define policy languages (e.g., REI [13]) for building security and privacy management systems in a pervasive computing environment.

In this paper, we present a set of ontologies that we have developed to support knowledge sharing and ontology reasoning in CoBrA. Our ontology (CoBrA ontology) is defined using the Web Ontology Language (OWL) [22]. The current version of the ontology (v0.2) models the basic concepts of people, agents, places and presentation events. It also describes the properties and relationships between these basic concepts which include (i) containment relationships between places, (ii) roles associated with people in presentation events, and (iii) typical intentions and desires of speakers and audience members (i.e., actions that a speaker or an audience intends to perform, and actions that a speaker or an audience desires others to achieve).

Unlike the designs of other ontologies such as Cyc [8] and SUO [17] that use formal methods to define a comprehensive collection of concepts for general purpose ontology modeling, our ontology design is aimed to create a set of ontological vocabularies with basic semantics that are pragmatic for building pervasive context-aware systems in the near future. We believe restricting an initial ontology design to what is implementable using the existing technologies is a necessary step in pushing ontology-driven design methodologies to pervasive computing.

The rest of this document is structured into five sections. In the next section, we overview CoBrA and discuss its design rationale. In Section 3, we discuss the role of the OWL ontology in the CoBrA system. Following our discussion, in Section 4, we describe the CoBrA ontology and how OWL is used to define this ontology. To show how CoBrA ontology may be used, a use case in the domain of an intelligent meeting room system is described in Section 5. In Section 6, we discuss research work that are closely related CoBrA. In the last section, we briefly discuss our future work and summarize this document.

2 Context Broker Architecture

CoBrA is an agent based architecture for supporting context-aware computing in intelligent spaces. Intelligent spaces are physical spaces (e.g., living rooms, vehicles, corporate offices and meeting rooms) that are populated with intelligent systems that provide pervasive computing services to users [14]. By context, we mean an understanding of a location, its environmental attributes (e.g., noise level, light intensity, temperature and motion) and the people, devices, objects and software agents it contains.

Central to our architecture is the presence of an intelligent *context broker* (or broker for short) that maintains and manages a shared model of contexts on the behalf of a community of agents. These agents can be applications hosted by mobile devices that a user carries or wears (e.g., cell phones, PDAs and headphones), services that are provided by devices in a room (e.g., projector service, light controller and room temperature controller) and web services that provide web presences for people, places and things in the physical world (e.g., services keeping track of people's and objects' whereabouts [15]).

In an intelligent space, the primary responsibilities of a broker are to (i) acquire contexts from heterogeneous information sources and maintain the consistency of the overall context knowledge through reasoning, (ii) help distributed agents to share context knowledge through the use of ontologies, agent communication languages and protocols, and (iii) protect the privacy of users by establishing and enforcing user-defined policies while sharing sensitive personal information with agents in the community.

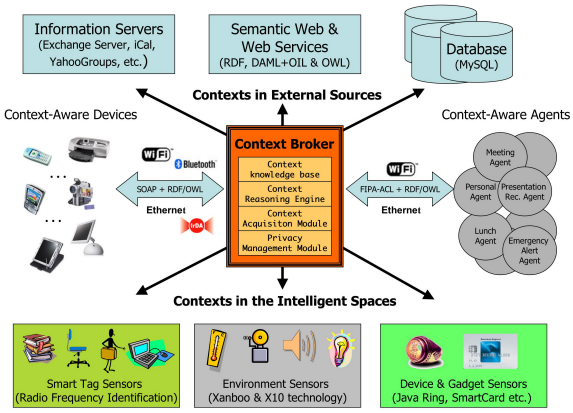

Fig. 1. In our approach an intelligent context broker acquires context information from devices, agents and sensors in its environment and fuses it into a coherent model, which is then shared with the devices and their agents.

A context broker has the following four main functional components:

1. **Context Knowledge Base**: a persistent storage space that stores context knowledge in an associated intelligent space. This knowledge base provides a set of API's for other components to assert, delete, modify and query stored knowledge.
2. **Context Reasoning Engine**: a reactive inference engine that reason over the knowledge that is stored in the knowledge base. Its main function is to deduce additional knowledge from information acquired from external sources and to maintain the consistency of the knowledge base.

3. **Context Acquisition Module**: a collection of pre-defined programming procedures for acquiring information from the external sources. It serves as a middleware abstraction for acquiring contexts from heterogeneous sources (e.g., physical sensors, web services, databases, devices and agents).
4. **Privacy Management Module**: a set of communication protocols and behavior rules that the broker follows when performing privacy management tasks (i.e., negotiate privacy policies with new users and enforcing these policies when sharing information with agents in the community).

2.1 An Intelligent Meeting Room Scenario

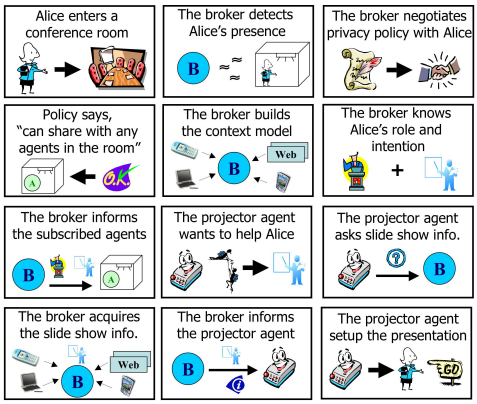

Fig. 2. The role of a context broker in an intelligent meeting room scenario

Figure 2 shows a typical scenario that demonstrates the key functions of a broker in an intelligent meeting room. As the person Alice enters a conference room, the broker acquires her presence information from various external sources (e.g., by detecting her RFID badge, sensing the proximity of the Bluetooth devices she carries and inferring the likelihood that she is participating in an on-going meeting). Before the broker can share any of her context information with agents in the room, the broker establishes privacy policies with Alice. Through a policy negotiation process, Alice permits the broker to share her context information with all agents in the room while she is present in the room.

As the broker acquires additional context information from other sources, the broker deduces the knowledge that Alice is the speaker of the meeting (her role) and is about to give a PowerPoint presentation (her intention). Immediately the broker informs all previously subscribed agents about Alice's role and intention. Among the recipients of this information, a projector agent believes it is in Alice's interest to automatically set up PowerPoint slides for her (her desire). The

projector agent subsequently queries the broker for the URL from which it can download Alice's presentation. The broker checks Alice's privacy policy to verify whether the URL information can be shared. If permission is granted, the broker replies with the necessary information. In case if no URL information exists in the knowledge base, the broker will execute appropriate context acquisition behavior and attempt to acquire it from external sources. Upon receiving a reply from the broker, the projector agent finishes the job of readying her slides for presentation.

2.2 Rationale and Design Issue

Our centralized broker design is motivated by the growing demand for intelligent mobile computing devices [9, 7, 4] and the increasing concern for privacy and security issues in sharing user information in an open and dynamic environment [1, 2]. In the past few years, computation speed and storage capacity in mobile devices have significantly increased and the trend is expected to continue for the foreseeable future.

However, due to size and form factor constraints, today's mobile devices still lack the necessary resources and facilities to become context-aware in comparison to their stationary counterparts (e.g., battery power, persistent storage and sensor access) [5]. In addition to resource issues, the growing popularity of wireless networks (e.g., Wi-Fi, Bluetooth and IrDA) raises great concerns for information security and personal privacy. For example, the network connectivity information associated a user device can give information about a user's location, the activities in which the user participates, and people whom the user is associated with [1, 2].

We believe that the introduction of a centralized context broker that operates on a stationary computer in an intelligent space can overcome resource issues in mobile devices and address privacy and security concerns. In our design, the burdens of acquiring and reasoning over context information will be shifted away from resource-limited mobile devices to a resource-rich server entity; the complications in establishing, monitoring and enforcing security, trust, and privacy polices will be simplified in the presence of a centralized manager.

Although the existence of context broker could bring about the above advantages, its centralized design could create a "bottle neck" situation in a distributed system, hindering the overall system performance. In our preliminary research work, we have not addressed this problem. However, according Kumar and Cohen [16], this "bottle neck" issue could be resolved through fault-tolerance by introducing a *persistent broker team*.

3 The Role of OWL in CoBrA

Our CoBrA ontology is defined using the OWL language. An explicit representation of the ontology underpins two important functions of a context broker. First, it provides a means for the broker to share context knowledge with agents

in an associated intelligent space. Second, it provides an ontology model which can help the broker to reason about contexts and detect knowledge inconsistency.

3.1 Enabling Knowledge Sharing

Knowledge sharing in pervasive context-aware systems requires all agents to share a common ontology[1]. To meet this requirement, previous systems [25, 3, 14, 19] often define ontologies based on *ad hoc* representation schemes (e.g., defining a set objects or data structures to represent shared knowledge). The key problem of this approach lays in system interoperability. In a pervasive computing environment, because devices, services and agents are likely to be independently developed without much pre-define agreements on how they should interoperate (i.e., on a priori identification of all those things one would want to communicate or discuss [11]), an *ad hoc* representation of the ontologies could limit agents' ability to interoperate.

In CoBrA, the representation of common ontologies is expressed in the form of web ontologies. Using the OWL language, ontology concepts are defined independent from any agent implementations, and their semantics are captured using standard knowledge representation vocabularies. Taking this approach, independently developed agents can share context knowledge with the broker without pre-defined agreements on how they should interoperate.

3.2 Providing Ontology Model for Context Reasoning

Context reasoning is a key function of the broker (see Section 2). Context reasoning involves deducing context knowledge from acquired situational information and detecting inconsistency in the knowledge base. To reason about contexts, the broker can exploit inference mechanisms that couple ontology models with general purpose logic inference engines, reasoning over the logic inferences that are allowed in the ontology model.

In previous systems (e.g., Intelligent Room [7], Context Toolkit [19] and Cooltown [15]), building context reasoning mechanisms have been difficult and costly [5]. In these systems, because the *ad hoc* representations of context knowledge are not suitable for building ontology models, they cannot be easily integrated with general purpose logic inference engines to enhance system "intelligence".

For this reason, we have explored an alternative approach that uses the OWL language to define ontologies. With the emergence of OWL reasoning engines (e.g., FaCT [12], RACER [23] and Bubo [24]), we believe defining ontologies using OWL is suitable for building context ontology models that can be easily integrated with logic inference engines, creating new opportunities for building more advanced intelligence systems in the future.

[1] This sharing might, in practice, be achieved with the help of ontology translation agents.

4 The CoBrA Ontology

This section describes key ontology concepts in the current version of the CoBrA ontology (v0.2)[2]. This ontology defines a set of vocabularies for describing people, agents, places and presentation events for supporting an intelligent meeting room system on a university campus. It also defines a set of properties and relationships that are associated with these basic concepts.

Figure 3 shows a complete list of the names of the classes and properties in the CoBrA ontology. In v0.2, there are 41 classes (i.e., RDF resources that are type of `owl:class`) and 36 properties (i.e., RDF resources that are type of either `owl:ObjectProperty` or `owl:DatatypeProperty`). These ontologies are expressed using the OWL/XML syntax [21].

CoBrA Ontology Classes		CoBrA Ontology Properties	
"Place" Related	Agents' Location Context	"Place" Related	Agent's Location Context
Place AtomicPlace CompoundPlace Campus Building AtomicPlaceInBuilding AtomicPlaceNotInBuilding Room Hallway Stairway OtherPlaceInBuilding Restroom Gender LadiesRoom MensRoom ParkingLot	ThingInBuilding SoftwareAgentInBuilding PersonInBuilding ThingNotInBuilding SoftwareAgentNotInBuilding PersonNotInBuilding ThingInRoom SoftwareAgentInRoom PersonInRoom	latitude longitude hasPrettyName isSpatiallySubsumedBy spatiallySubsumes accessRestricted-ToGender lotNumber	locatedIn locatedInAtomicPlace locatedInRoom locatedInRestroom locatedInParkingLot locatedinCompoundPlace locatedInBuilding locatedInCampus
"Agent" Related	Agent's Activity Context	"Agent" Related	Agent's Activity Context
Agent Person SoftwareAgent Role SpeakerRole AudienceRole IntentionalAction ActionFoundInPresntation	PresentationSchedule EventHappeningNow PresentationHappeningNow RoomHasPresentationHappeningNow ParticipantOfPresentation-HappeningNow SpeakerOfPresentationHappeningNow AudienceOfPresentationHappeningNow PersonFillsRoleInPresentation PersonFillsSpeakerRole PersonFillsAudienceRole	hasContactInformation hasFullName hasEmail hasHomePage hasAgentAddress fillsRole isFilledBy intendsToPerform desiresSomeone-ToAchieve	participatesIn startTime endTime location hasEventHappeningNow invitedSpeaker expectedAudience presentationTitle presentationAbstract presentation eventDescription eventSchedule

Fig. 3. A complete list of the names of the classes and properties in the CoBrA ontology (v0.2).

Our ontology is categorized into four distinctive but related themes: (i) concepts that define physical places and their associated special relationships (e.g., containment relationship, social and organizational properties)[3], (ii) concepts that define agents (i.e., both human agents and software agents) and their associated attributes, (iii) concepts that describe the location contexts of an agent on a university campus, and (iv) concepts that describe the activity contexts

[2] A complete version of the ontology is available at `http://daml.umbc.edu/ontologies/cobra/0.2/cobra-ont`

[3] In v0.2, only containment relationship is defined, additional properties will be included in the next version of the ontology.

of an agent, including the roles of speakers and audiences and their associated desires and intentions in a presentation event. In the rest of this section, we will discuss each of these four themes.

4.1 Concepts Related To Places

The notion of a place in CoBrA is restricted to a set of physical locations that are typically found on a university campus. These locations include *campus*, *building*, *room*, *hallway*, *stairway*, *restroom*, and *parking lot*. These physical locations are all assumed have well-defined spatial boundaries (e.g., all locations can be uniquely identified by geographical coordinates – longitude and latitude). In addition, all locations on a university campus have identifiable string names that are assigned to them by some official bodies (e.g., by the university administration).

When modeling physical locations, we define a class called `Place` which generalizes all type of locations on a campus. This abstract class defines a set of properties that are common to all concrete physical location classes, which consists of `longitude`, `latitude` and `hasPrettyName`.

`Place` classes (including subclasses) have associated containment relationships. These relationships are defined by two related object properties[4] called `spatiallySubsumes` and `isSpatiallySubsumedBy`. The former describes the subject of this property spatially subsumes the object of this property (e.g., a building spatially subsumes a room in the building), and the latter describes the subject of this property is spatially subsumed by the object of this property (e.g., a room in the building is spatially subsumed by the building). In the context of the OWL language, these two properties are defined as an inverse property of each other.

Note that in the current version of the ontology, the domain and the range of both `spatiallySubsumes` and `isSpatiallySubsumedBy` properties are of the class type `Place`. In other word, these two properties cannot be used to make statements about the containment of a person or an agent in a physical place. However, in Section 4.3, we will describe alternative constructs for expressing this type of statements.

In addition to containment relationships, physical places may be also associated with events and activities (e.g., a meeting may be taken place in a room, or an annual festive may be taken place on a university campus). In order to make statements about some events that are currently associated with a particular place, we introduce an additional object property called `hasEventHappeningNow`. The domain and range of this property are of the class `Place` and the class `EventHappeningNow`, respectively. The `EventHappeningNow` class represents a set of all events that are currently taking place (details of this class is discussed in Section 4.5).

[4] This refers to the `owl:ObjectProperty` property

```
<owl:Class rdf:ID="Place">
  <owl:unionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#AtomicPlace"/>
    <owl:Class rdf:about="#CompoundPlace"/>
  </owl:unionOf>
  …
</owl:Class>

<owl:Class rdf:ID="AtomicPlace">
  <rdfs:subClassOf rdf:resource="#Place"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty
          rdf:resource="#isSpatiallySubsumedBy"/>
      <owl:allValuesFrom
          rdf:resource="#CompoundPlace"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty
          rdf:resource="#spatiallySubsumes"/>
      <owl:cardinality>0</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <owl:unionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#AtomicPlaceInBuilding"/>
    <owl:Class
       rdf:about="#AtomicPlaceNotInBuilding"/>
  </owl:unionOf>
</owl:Class>
```

```
<owl:Class rdf:ID="CompoundPlace">
  <rdfs:subClassOf rdf:resource="#Place"/>
  <owl:disjointWith rdf:resource="#AtomicPlace"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty
          rdf:resource="#isSpatiallySubsumedBy"/>
      <owl:allValuesFrom
          rdf:resource="#CompoundPlace"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty
          rdf:resource="#spatiallySubsumes"/>
      <owl:allValuesFrom
          rdf:resource="#Place"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <owl:unionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#Campus"/>
    <owl:Class rdf:about="#Building"/>
  </owl:unionOf>
</owl:Class>
```

Fig. 4. A partial ontology definition of the `AtomicPlace` & `CompoundPlace` classes in OWL/XML syntax

AtomicPlace From the list of concrete physical locations that we have mentioned (i.e., campus, building, room, hallway, stairway, etc.), some of these locations usually do not contain (spatially subsume) other physical locations. For example, hallways, stairways and rooms in a building usually are not usually considered to be a type of physical place that contains other places.

For this reason, we introduce an abstract class called `AtomicPlace` to represent a set of all physical places that do not contain other physical places. This class inherits all properties from its superclass `Place`. However, it puts restrictions on the range of the two properties `spatiallySubsumes` and `isSpatiallySubsumedBy`. In this `AtomicPlace` class, the cardinality of the property `spatiallySubsumes` is 0, indicating all instances of this class do not contain any other physical places. On the other hand, the range of the property `isSpatiallySubsumedBy` is restricted to the class `CompoundPlace`, which is a subclass of `Place`. This `CompoundPlace` class represents all physical places that may contain other physical places. Figure 4 shows partial representation of these classes in OWL/XML syntax.

Some subclasses of the `AtomicPlace` class include `Room`, `Hallway`, `Stairway`, `Restroom`, `LadiesRoom`, `MensRoom` and `ParkingLot`.

4.2 CompoundPlace

While the `AtomicPlace` class is introduced to represent a set of places that contains zero number of `Place` instances, the `CompoundPlace` class is defined to represent a set of places that contains at least one or more numbers of `Place` instances. This class is also a subclass of `Place`.

Being a subclass of the `Place` class, `CompoundPlace` inherits all properties from its parent class. In order to express all instances of the `CompoundPlace` class should only be spatially subsumed by instances of other `CompoundPlace`, the range of this class's property `isSpatiallySubsumedBy` is restricted to have class type `CompoundPlace`. This restriction excludes all instances of the `CompoundPlace` class to be spatially subsumed by instances of the `AtomicPlace`.

4.3 Concepts Related To Agents

The notion of an agent in CoBrA represents both humans agents and software agents. Human agents are simply users in an intelligent space. Software agents, on the other hand, are autonomous computing entities that provide services to users (either directly or indirectly) in an associated space.

All agents have associated properties that describes their contact information, which includes uniquely identifiable names, URL's to their home pages, and email addresses. In addition, agents are assumed to have certain roles in different events and activities (e.g., a person can have the speaker role in a presentation event, and device agents in the close vicinity may take on the presentation assistant role during the presentation session). Different roles may give rise to different desires and intentions of an agent.

In the CoBrA ontology, the notions of desire and intention are both associated with actions[5]. Specifically, the notion of desire is defined as an agent's desire for some actions to be achieved by other agents (e.g., a person with the speaker role may desire some service agents to dim the lights when his presentation starts), and the notion of intention is defined as an agent's intend to perform some particular actions (e.g., a person with the audience role may intend to download a copy of the slides after attending a presentation event).

To begin our ontology modeling for agents, we introduce a general class called `Agent`, which is a set of all human agents and computational agents. We define the class `Person` to represent human agents and the class `SoftwareAgent` to represent computational agents (both of which are subclasses of the `Agent` class and disjoints with each other). All agents in our ontology are associated with properties that describe their contact information. To generalize properties that serve as descriptions of contact information, we define an object property called `hasContactInformation`. From this property, we further define sub-properties of contact information, which consist of `hasFullName`, `hasEmail`, `hasHomePage` and `hasAgentAddress`.

Role In our ontology, the class `Role` represents a set of all roles that can be associated with an agent. In other words, it is an abstract class that generalizes all possible types of agent roles in the CoBrA ontology. In v0.2 of the ontology, pre-defined subclasses of `Role` consist of `SpeakerRole` and `AudienceRole`.

[5] the semantics of an action is not formal defined in the current version of the ontology. In v0.2, all instances of actions are assumed to be atomic.

To associate roles with an agent, the object properties `fillsRole` and `isFilledBy` are defined. In the context of the OWL language, these two properties are inverse property of each other – `fillsRole` has domain `Agent` and range `Role`, and `isFilledBy` has domain `Role` and range `Agent`.

```
<owl:Class rdf:ID="Role"/>

<owl:ObjectProperty rdf:ID="fillsRole">
  <rdfs:domain rdf:resource="#Agent"/>
  <rdfs:range rdf:resource="#Role"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="isFilledBy">
  <owl:inverseOf rdf:resource="#fillsRole"/>
</owl:ObjectProperty>

<owl:Class rdf:ID="SpeakerRole">
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#RoleInPresentation"/>
    <owl:Restriction>
      <owl:onProperty
rdf:resource="#intendsToPerform"/>
      <owl:hasValue
rdf:resource="#GivePPTPresentation"/>
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>

<owl:Class rdf:ID="AudienceRole">
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#RoleInPresentation"/>

    <owl:Restriction>
      <owl:onProperty
         rdf:resource="#intendsToPerform"/>
      <owl:hasValue
         rdf:resource="#AttendPresentation"/>
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
```

```
<owl:Class rdf:ID="IntentionalAction"/>

<owl:Class rdf:ID="ActionFoundInPresentation">
  <rdfs:subClassOf rdf:resource="#IntentionalAction"/>

  <owl:oneOf rdf:parseType="Collection">
    <owl:Thing rdf:about="#GivePPTPresentation"/>
    <owl:Thing rdf:about="#SetupPPTPresentation"/>
    <owl:Thing rdf:about="#AdjustLightIntensity"/>
    <owl:Thing rdf:about="#DistributeHangouts"/>
    <owl:Thing rdf:about="#AttendPresentation"/>
    <owl:Thing rdf:about="#AcquireHangouts"/>
    <owl:Thing rdf:about="#ExchangeContact"/>
  </owl:oneOf>
</owl:Class>

<owl:ObjectProperty rdf:ID="intendsToPerform">
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
                <owl:Class rdf:about="#Role"/>
                <owl:Class rdf:about="#Agent"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
  <rdfs:range rdf:resource="#IntentionalAction"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="desiresSomeoneToAchieve">
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
                <owl:Class rdf:about="#Role"/>
                <owl:Class rdf:about="#Agent"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
  <rdfs:range rdf:resource="#IntentionalAction"/>
</owl:ObjectProperty>
```

Fig. 5. This is a partial definition of the concepts related to roles, intentions and desires in an intelligent meeting room system.

Intentional Actions All actions in CoBrA are defined as instances of the class `IntentionalAction`. Informally, intentional actions are actions that an agent performs intentionally and with certain goals in mind. In our design, we assume domain applications will extend this class to define specialized subclasses and instances. To support the construction of intelligent meeting room systems, we have pre-defined a set of concrete instances of `IntentionalAction` that are common in presentation events (see Figure 5).

All instances of the `IntentionalAction` class (or its subclasses) can be associated with either an instance of the `Role` class or the `Agent` class through object properties `intendsToPerform` or `desiresSomeoneToAchieve`. The domain of these two properties are defined to be a union of the class `Role` and `Agent` (see Figure 5).

4.4 Concepts Related to Agent's Location Context

In the last two sections, we have described a set of CoBrA ontology concepts for physical locations and agents. In this section, we will discuss additional concepts for modeling the location context of agents.

By location context, we mean a collection of dynamic knowledge that describes the location of an agent. In the context of the OWL language, this knowledge is a collection of RDF statements that describes the location property of an agent. To model the location property of an agent, we have introduced an object property called `locatedIn`, which has range `Place`[6].

Physical locations, as we have discussed previously in Section 4.1, are categorized into two distinctive classes namely `AtomicPlace` (e.g., hallways and rooms) and `CompoundPlace` (e.g., campus and building). Following the semantics of these two classes, *no agent can locate in two different atomic places at the same time, but any agent can locate in two different compound places at the same time*. In the context of the OWL language, any two instances of the `AtomicPlace` class are different if and only if both instances have distinctive object values[7] for the same class property.

To capture the notion an agent can be in an atomic and a compound place, from the `locatedIn` property we define two sub-properties called `locatedInAtomicPlace` and `locatedInCompoundPlace`. The former restricts its range to the `AtomicPlace` class, and the latter restricts its range to the `CompoundPlace` class. From these two properties, we can define additional properties that further restricts the type of physical place an agent is located in. For example, `locatedInRoom`, `locatedInRestroom` and `locatedInParkingLot` are sub-properties of `locatedInAtomicPlace`; `locatedInCampus` and `locatedInBuiding` are sub-properties of `locatedInCompoundPlace`.

Since all agents in CoBrA are associated with different types of location properties, we can generalize subsets of these agents in according to their location properties. To do so, we define `PersonInBuilding` and `SoftwareAgentInBuilding` to represent a set of people and software agents who are located in a building, respectively. The complement of these classes are `PersonNotInBuilding` and `SoftwareAgentNotInBuilding`.

4.5 Concepts Related to Agent's Activity Context

The activity context of an agent, similar to the location context, is a collection of dynamic knowledge about certain aspects of an agent's situational condition. While location context describes the location at which the agent is situated, activity context describes activities in which the agent participates. In the current version of the ontology, the notion of an activity is restricted to represent a set of all typical group activity events in a meeting room (meeting, presentation and discussion)[8].

Activity events are assumed have schedules. For presentation events, we have defined `PresentationSchedule` class to represent their schedules. Presentation

[6] The domain of this property is `owl:Thing`, indicating any thing may be located in some physical place.

[7] an object value refers to the object in an N-triple statement (i.e. (<subject>, <predicate>, <object>)

[8] In v0.2 of the ontology, we have only included concepts related to presentation events. In the future version, we will extend the ontology to includes other activity events

schedules are defined to have `startTime`, `endTime` and `location` properties, and each of which respectively represents the start time of a presentation, the end time of a presentation and the location of a presentation event. Each presentation event has one or more invited speaker and expected audience. These two concepts are defined using the `invitedSpeaker` and `expectedAudience` properties. In addition to start time, end time and location, the schedule of a presentation usually includes a title and an abstract of the presentations. To model these two concepts, we introduce `presentationTitle` and `presentationAbstract` properties.

The activity context of an agent is usually associated with activity events that are currently happening. For example, the activity context of a speaker includes the presentation event that he/she is giving the presentation at. To model this, we introduce the `PresentationEventHappeningNow` class. This class is a subclass of the `EventHappeningNow` class which models an event with the time predicate "now".

For a given presentation that is currently happening, we can specialize the type of rooms at which the event takes place. For example, a room that has an on-going presentation event is defined as `RoomHasPresentationEventHappening-Now`, which is a subclass of `Room` and restricts the range of its `hasEvent-HappeningNow` property to the class `PresentationSchedule`. In addition, we can also specialize people who has various roles in an on-going event. For example, a set of all people who have the speaker role of some on-going presentation event is defined as the `SpeakerOfPresentationHappeningNow` class. Similarly, we define the `AudienceOfPresentationHappeningNow` class to represent a set of all people who have the audience role of some on-going presentation event.

5 Ontology Use Case

In the last section, we have described the key ontology concepts in the CoBrA ontology. In this section, we describe a use case that shows how CoBrA ontology can be used to reason about the location context of a person.

In a pervasive context-aware environment, we presume people presence sensors are embedded in the physical environment. These sensors exploit various sensing techniques such as RFID, Bluetooth, and voice/facial recognitions to detect the presence of people. As people wonders around in a physical space, his/her presence will be detected by these sensors. However, this information may be lack of sufficient details to support context-aware computing because sensors usually have relatively limited capability in knowledge representation. For this reason, context brokers could be used to enrich information acquired by the sensors through ontology reasoning.

For example, sensors in a room detect the presence of a person. They encode this knowledge using the CoBrA ontology and send it to the context broke (see Part A of Figure 6). By reasoning over the containment relationship between different `Place` entities, the context broker can deduce additional location contexts about the person (see Part B of Figure 6).

Part A: Contexts send to the Context Broker from Sensors

```
<cobra:Person rdf:about="http://umbc.edu/~hchen4">
  <cobra:locatedInRoom rdf:resource="http://www.cs.umbc.edu/ont/building#ECS210I"/>
</cobra:Person>
```

Part B: Additional contexts inferred by the Context Broker

```
<cobra:Person rdf:about="http://umbc.edu/~hchen4">
  <cobra:locatedInAtomicPlace rdf:resource="http://www.cs.umbc.edu/ont/building#ECS210I"/>
  <cobra:locatedIn rdf:resource="http://www.cs.umbc.edu/ont/building#ECS210I"/>
  <cobra:locatedInBuilding rdf:resource="http://www.umbc.edu/ont/campus-map#Building-ECS"/>
  <cobra:locatedOnCampus rdf:resource="http://www.umbc.edu/ont/campus-map#UMBC-Main"/>
</cobra:Person>

<cobra:ThingInBuilding rdf:about="http://umbc.edu/~hchen4">
  <cobra:locatedIn rdf:resource="http://www.cs.umbc.edu/ont/building#ECS210I"/>
</cobra:ThingInBuilding>

<cobra:PersonInBuilding rdf:about="http://umbc.edu/~hchen4">
  <cobra:locatedIn rdf:resource=
  "http://www.cs.umbc.edu/ont/building#ECS210I"/>
</cobra:PersonInBuilding>
```

Fig. 6. Part A: after detecting the RFID badge that Harry Chen wears, the sensors in Room ECS210I informs the context broker of his presence. Part B: Through ontology reasoning, the broker can deduce additional properties about a person and classify that person in according the containment relationship of the `Place` ontology

6 Related Work

Our work is closely related to other pervasive and context-aware computing research such as Intelligent Room [7], Context Toolkit [19] and Cooltown [15], One.World [10] and Centaurus [14]. In comparison to the previous systems, our novel design of the context broker attempts to address challenging issues such as developing explicit ontology representations of contexts, supporting context reasoning and maintenance through logic inferences and providing user privacy protection using policies (also see discussions in Section 2.2).

In the previous systems, user location contexts are widely used for guiding the decision making of context-aware applications [19, 7, 14]. However, none of them have explored the space and spatial relationship aspects of the location contexts (i.e., information that describes the whole physical space that surrounds a particular location and its relationship to other locations). Modeling space and spatial relationships are important in CoBrA. We currently have a simple model of space and spatial relationships (see Section 4.1). As our ontology and needs evolve, we will make it richer and more sophisticated.

At the moment, there are two distinctive versions of spatial ontologies namely the spatial ontology in SUO [17] and the upper Cyc ontology [8]. In the DAML+OIL community, recent discussions on the daml-spatial mailing list have initiated the work to develop a Semantic Web version of the spatial ontology based the SUO and Cyc[9]. In the future, we plan on using, if possible, or at least mapping to, if feasible, one of these consensus ontologies for space.

[9] `http://www.daml.org/listarchive/daml-spatial/`

7 Conclusion and Future work

An explicit representation of ontologies plays two important roles in building pervasive context-aware systems: (i) enabling knowledge sharing and (ii) providing ontology model for context reasoning. In this paper, we have described the Context Broker Architecture and ontologies for supporting knowledge sharing and ontology reasoning in this system.

At the present, we are developing a reasoning engine called F-OWL[10] to support logic inference over OWL and the CoBrA ontologies. Our prototype system is implemented using an F-logic based language called Flora-2 [18]. The Flora-2 system translates a unified language of F-logic, HiLog, and Transaction Logic into the XSB deductive engine [27]. The language syntax of Flora-2 is similar to TRIPLE [20], and they both allow ontology semantics to defined using rules. An initial prototype of this reasoning engine is expected to be available in the end of May 2003.

As a part of our long term research plan, we will prototype an intelligent context broker and integrate this broker with the Centaurus system (a framework for building pervasive computing services developed at UMBC) [14]. Our goal is to create and deploy a pervasive context-aware meeting room in the newly constructed Information Technology and Engineering Building on the UMBC main campus[11].

References

1. M. Ackerman, T. Darrell, and D. J. Weitzner. Privacy in context. *Special Issue on Context-Aware Computing. Human-Computer Interaction*, 16(2-4), 2001.
2. V. Bellotti and A. Sellen. Design for privacy in ubiquitous computing environments. In *Proceedings of the Third European Conference on Computer Supported Cooperative Work (ECSCW'93)*, pages 77–92. Kluwer, 1993.
3. F. Bennett, T. Richardson, and A. Harter. Teleporting - Making Applications Mobile. In *Proceedings of 1994 Workshop on Mobile Computing Systems and Applications*, Santa Cruz, December 1994.
4. G. Chen and D. Kotz. A survey of context-aware mobile computing research. Technical Report TR2000-381, Dartmouth College, Computer Science, Hanover, NH, nov 2000.
5. H. Chen. An intelligent broker architecture for context-aware systems. PhD. dissertation proposal. See `http://users.ebiquity.org/~hchen4/phd/hcthsisp.pdf`, 2003.
6. H. Chen, S. Tolia, C. Sayers, T. Finin, and A. Joshi. Creating context-aware software agents. In *Proceedings of the First GSFC/JPL Workshop on Radical Agent Concepts*, 2001.
7. M. H. Coen. Design principles for intelligent environments. In *AAAI/IAAI*, pages 547–554, 1998.
8. Cyccorp Inc. *The Upper Cyc Ontology*, 1997. `http://www.cyc.com/cyc-2-1/cover.htm`.

[10] F-OWL web site: `http://users.ebiquity.org/~hchen4/fowl`

[11] ITE building construction live feed: `http://www.cs.umbc.edu/ITE/ITE.html`

9. M. Dertouzos. *The Unfinished Revolution*. HarperCollins Publishers, 2001.
10. R. Grimm, T. Anderson, B. Bershad, and D. Wetherall. A system architecture for pervasive computing. In *Proceedings of the 9th ACM SIGOPS European Workshop*, pages 177–182, 2000.
11. J. Heflin. Web ontology language (owl) use cases and requirements, 2003.
12. I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In H. Ganzinger, D. McAllester, and A. Voronkov, editors, *Proceedings of the 6th International Conference on Logic for Programming and Automated Reasoning (LPAR'99)*, number 1705 in Lecture Notes in Artificial Intelligence, pages 161–180. Springer-Verlag, 1999.
13. L. Kagal, T. Finin, and A. Joshi. A policy language for a pervasive computing environment. In *IEEE 4th International Workshop on Policies for Distributed Systems and Networks*, 2003.
14. L. Kagal, V. Korolev, H. Chen, A. Joshi, and T. Finin. Centaurus : A framework for intelligent services in a mobile environment. In *Proceedings of the International Workshop on Smart Appliances and Wearable Computing (IWSAWC)*, 2001.
15. T. Kindberg and J. Barton. A Web-based nomadic computing system. *Computer Networks (Amsterdam, Netherlands: 1999)*, 35(4):443–456, 2001.
16. S. Kumar, P. R. Cohen, and H. J. Levesque. The adaptive agent architecture: Achieving fault-tolerance using persistent broker teams. In *Proceedings of the Fourth International Conference on Multi-Agent Systems*, pages 159–166, 2000.
17. I. Niles and A. Pease. Towards a standard upper ontology. In *Proceedings of the 2nd International Conference on Formal Ontology in Information Systems (FOIS-2001)*, 2001.
18. K. Sagonas, T. Swift, D. S. Warren, J. Freire, P. Rao, B. Cui, and E. Johnson. *The XSB Programmers' Manual*, 2002.
19. D. Salber, A. K. Dey, and G. D. Abowd. The context toolkit: Aiding the development of context-enabled applications. In *CHI*, pages 434–441, 1999.
20. M. Sintek and S. Decker. Triple—a query, inference, and transformation language for the semantic web. In *Proceedings of International Semantic Web Conference (ISWC-02)*, 2002.
21. M. K. Smith, C. Welty, and D. McGuinness. Owl web ontology language guide. `http://www.w3.org/TR/owl-guide/`, 2003.
22. F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein. Owl web ontology language reference. `http://www.w3.org/TR/owl-ref/)`, 2002.
23. R. M. Volker Haarslev. Description of the racer system and its applications. In *Proceedubgs International Workshop on Description Logics (DL-2001)*, 2001.
24. R. Volz, S. Decker, and D. Oberle. Bubo - implementing owl in rule-based systems. 2003.
25. R. Want, B. Schilit, N. Adams, R. Gold, K. Petersen, D. Goldberg, J. R. Ellis, and M. Weiser. An overview of the PARCTAB ubiquitous computing experiment. *IEEE Personal Communications*, 2(6):28–33, Dec 1995.
26. M. Weiser. The computer for the 21st century. *Scientific American*, 265(30):94–104, 1991.
27. G. Yang and M. Kifer. *Flora-2: User's Manual*. Department of Computer Science, Stony Brook University, Stony Brook, 2002.