An Ontology for Context Aware Pervasive Computing Environments[1]

Harry Chen and Tim Finin
University of Maryland, Baltimore County
Baltimore MD 21250 USA
{hchen4,finin}@umbc.edu

Abstract

We describe a framework for an agent based pervasive computing environment. Central to our framework 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 web describing the space and events and activities scheduled for it. The context broker integrates and reasons over this information 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 undergird the communication and representation.

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 the service, acting on our behalf, and delivering services in an anywhere, anytime fashion [6, 7, 9, 15]. Characteristic of this vision is the use of wireless networking, sensor rich environments, mobile and wearable computing, and intelligent human-computer interfaces. 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. At UMBC, we are developing a new pervasive context-aware computing infrastructure called Context Broker Architecture (CoBrA), to support ubiquitous agents, services and devices to behave intelligently in according to their situational contexts.

By context, we mean an understanding of a location and its environmental attributes and the people, devices, objects and software agents it contains. This understanding necessarily extends to modeling the activities and tasks going on as well as some understanding of the beliefs, desires, commitments, and intentions of the human and artificial agents involved. This infrastructure will acquire and reason about contexts, helping resource-poor agents and devices maintain consistent contextual knowledge and coordinate, and cooperate. It will enforce declared security and privacy policies, and provide intelligent interfaces and wearable computers with the necessary context in which to operate.

A key to realizing this architecture is the use of a set of common ontologies that undergird the communication and representation. In this document, we will describe our research work in building this set of ontologies using Web Ontology Language (OWL) [2]. The rest of this document is structured as the following:

- Section 2 gives an overview of the CoBrA architecture and its design rationale.
- Section 3 describes the CoBrA ontology and briefly overviews the OWL language.
- Section 4 discusses use cases and how ontology reasoning can be defined to build an intelligent meeting room system.
- Section 5 summarizes this document and describes our future work.

2 Context

Context is any information that can be used to characterize the situation of a person, a computing device, or a software agent. In an intelligent space (i.e., conference room, office room, and living room) [7], context

[1] This paper is a draft submitted to the IJCAI 03 Workshop on Ontologies and Distributed Systems This work was partially supported by DARPA contract F30602-97-1-0215 and Hewlett Packard.

can be defined as information that characterizes the identity and attributes of people, devices and agents in the space. This information can include specific locations, capabilities and services offered and sought, the activities and tasks in which they are engaged, and situational roles, beliefs, desired and intentions. It must also include information about the physical environment (e.g. lighting, noise levels, movement) since this may change the way users interact with any devices they may be using. Context-aware computing offers many advantages, allowing systems to act more autonomously and take initiative, but informed by a better model of what their users need and want [6]. Building and deploying context aware systems in open, dynamic environments raises a new set of research challenges which we will address and solve, integrating results to realize an intelligent space that senses, learns, reasons and acts to construct an actionable understanding of a space. Figure 1 shows a typical scenario from an intelligent meeting room system.

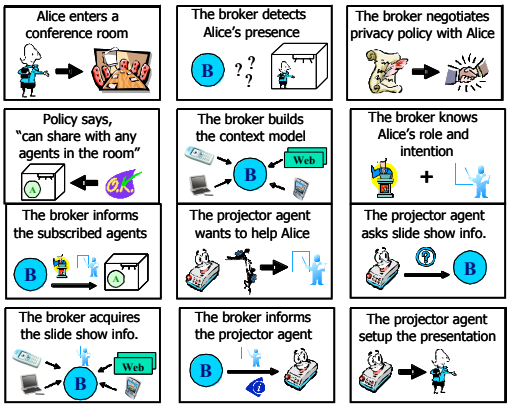

Figure 1. A typical scenario from an intelligent briefing

The context broker. Our work will be built around a broker-centric agent architecture to support context-aware computing in intelligent spaces. In this architecture, a broker will maintain and manage a shared context model on the behalf of a community of devices and agents, provide necessary common services including ontology mediation and teamwork support, and enforce policies for security, trust and privacy. The broker has the following four responsibility: (1) to acquire situational information from heterogeneous sources (e.g., from the Web, corporate databases, agents and devices); (2) to reason about the context in an intelligent space by interpreting acquired situational information; (3) to help distributed agents to share contextual knowledge through high-level agent communications; (4) to protect the privacy of users by establishing and enforcing user-defined policies when sharing contextual knowledge with agents in the community; (5) to coordinate agents to work together as a team to achieve shared objectives; and (6) to maintain contextual knowledge on the behalf of resource-poor agents and devices (i.e., detecting and resolving inconsistency and ambiguous contextual knowledge). In addition, this architecture will provide ontologies for describing various types of contexts in the intelligent spaces and policy languages that allow users to define policy rules for controlling the access and sharing of their personal contextual information.

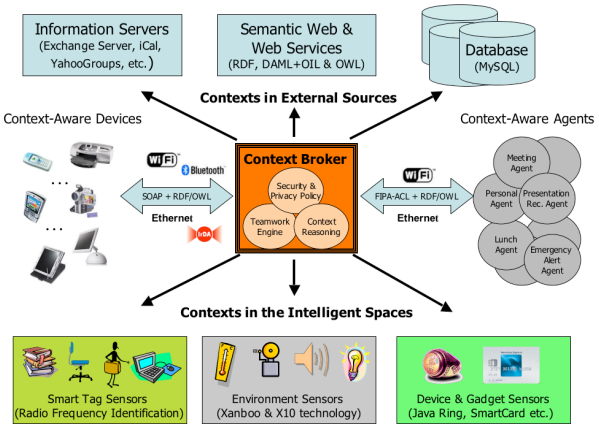

Figure 2. In our approach an intelligent context broker acquires context information from devices, agents and sensors in its environment and fuses it into a coherent context model, which is then shared with the devices and their agents.

The rationale behind a centralized design of the broker is motivated by the growing demand of context-aware agents and devices that operate on network-enabled computing devices (e.g., Bluetooth-enabled PDAs, cell phones, and printers). Because these devices have limited computing resources, agents often cannot completely depend on the hosting devices to support their context-aware behaviors. When taking a centralized approach, the burdens of acquiring and reasoning about contexts are shifted away from resource-limited agents to a resource-rich server agent. A broker-centric design also enables the monitoring and enforcement of security, trust and privacy policies, assuming no agent is allowed to share any user information without proper permissions from the broker (see Figure 2).

Our initial design for a context broker has five functional components. The knowledge base defines the ontologies of the intelligent meeting domain and heuristics domain knowledge (e.g., the office hours of a company are usually from 9:00 AM to 5:00 PM, and no one person can be physically present in two differ-

ent meeting rooms at the same time). The reasoning engine is responsible for reasoning with *ontology knowledge* and *contextual knowledge*. The ontology knowledge is static knowledge derived from the underlying ontology model (in OWL/RDF [1]), and the contextual knowledge is dynamic knowledge that is inferred from acquired situational information. The inference engine also has the function of detecting and resolving inconsistent knowledge using domain heuristics and probability. For example, if a person is thought to be present in a conference room after a meeting has ended, the daily schedule of that person can be used to detect if that person is actually in the room or is participating in another event at some other location. In addition, the inference engine will apply learning algorithms and pattern recognition mechanisms to learn about non-uniform and inconsistent user behaviors.

The context acquisition module is a collection of pre-defined programming modules for acquiring situational information from heterogeneous sources, for example, from hardware sensors and from Semantic Web services. These modules serve as middleware abstractions to help the broker to acquire situational information. The teamwork coordination module has a set of pre-defined plans and a reasoning engine to coordinate teamwork. For example, as a speaker enters a conference room to give a presentation, the broker instructs the projector agent and video camera agent to act as a team to help the speaker set up PowerPoint slides and recording his presentation. By instructing each team member to follow a shared team plan, agents can be coordinated to avoid taking conflicting actions (e.g., both projector agent and the video camera agent want to take control of the light control in the room, but one is interested to dim the light and other is interested to brighten the light).

The behavior component is a set of communication protocols that the broker follows when interact with users and agents in the system. Before a broker can share a user's information with agents in the community, the broker must establish a privacy policy with that user. The *privacy policy negotiation protocol* is defined for that purpose. After a policy has been established, each rule in the policy either grants or denies the sharing of certain user information with a specific type of agent.

3 The COBRA Ontology

In order to realize the Context Broker architecture, we need to develop a set of common ontologies for enabling knowledge sharing and ontology reasoning. In our preliminary research, we have developed an ontology for modeling contexts using the OWL language.

OWL is one of the emerging Semantic Web languages that are endorsed by the W3C for building ontologies [1, 2]. In the Semantic Web vision, OWL can help web services and agents to share information and interoperate. Using OWL, one can 1) formalize a domain by defining classes and properties of those classes, 2) define individuals and assert properties about them, and 3) reason about these classes and individuals [1]. The OWL language builds on the DAML+OIL language [12] and both are layered on top of on the standard RDF/RDFS triple data model (i.e., subject, predicate, and object) [16].

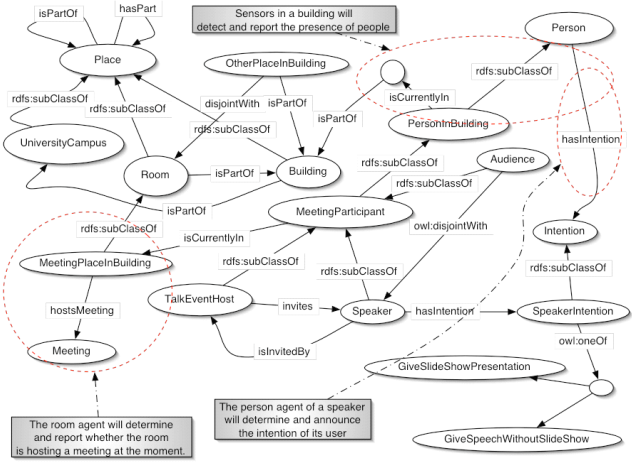

Figure 3. This is a graph representation of the CoBrA ontology. Each oval with solid line represents an OWL class. Each oval with broken line indicates the kind of information that a Context Broker will receive from other agents and sensors in the environment.

The current version of the CoBrA ontology[2] (version 0.1) consists of 17 classes and 32 properties definitions (Figure 3 shows a graph representation of some of the key CoBrA ontology definitions). This ontology defines some of the common relationships and attributes that are associated with people, places and activities in an intelligent space. This version of the CoBrA ontology is defined using the XML syntax because representing triples data model in XML helps computing machines to process semantic ontologies (Fig-

[2] http://daml.umbc.edu/ontologies/0.1/cobra

ure 4 shows a part of the CoBrA ontology in XML syntax). In the future version, we will provide support for an alternative syntax that is more human-readable (e.g., Notation-3 [3]).

```
<owl:Class rdf:ID="Place">
   <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasFullAddress"/>
      <owl:cardinality>1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasAliasName"/>
      <owl:minCardinality>1</owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasPart"/>
      <owl:allValuesFrom rdf:resource="#Place"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#isPartOf"/>
      <owl:allValuesFrom rdf:resource="#Place"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:ID="PersonInBuilding">
  <rdfs:subClassOf rdf:resource="#Person"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#isCurrentlyIn"/>
      <owl:allValuesFrom>
         <owl:Restriction>
         <owl:onProperty rdf:resource="#isPartOf"/>
         <owl:allValuesFrom rdf:resource="#Building"/>
         </owl:Restriction>
      </owl:allValuesFrom>
      <owl:maxCardinality>1</owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:ID="MeetingParticipant">
  <rdfs:subClassOf rdf:resource="#PersonInBuilding"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#isCurrentlyIn"/>
      <owl:allValuesFrom
rdf:resource="#MeetingPlaceInBuilding"/>
      <owl:maxCardinality>1</owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:ID="SpeakerIntention">
  <rdfs:subClassOf rdf:resource="#Intention"/>
  <owl:oneOf rdf:parseType="Collection">
    <owl:Item rdf:about="#GiveSlideShowPresentation"/>
    <owl:Item rdf:about="#GiveSpeechWithoutSlideShow"/>
  </owl:oneOf>
</owl:Class>
```

Figure 4. This is a part of the CoBrA ontology represented in XML syntax. The complete ontology is available at `http://daml.umbc.edu/ontologies/0.1/cobra`

The key top-level ontology of CoBrA consists of classes and properties that describe "Person", "Place", and "Intention". Each class defines the most general properties of the concept that it models. In order to achieve this using the OWL language, each class is defined to be a subclass of a set of anonymous classes, and each anonymous class restricts some of its properties. For example, in Figure 4, the "Place" class is defined as a subclass of four anonymous classes that each of which restricts one of the class properties, "hasFullAddress", "hasAliasName", "isPartOf" and "hasPart".

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

- The "PersonInBuilding" class which defines a type of people who are currently in a building,
- The "MeetingParticipant" class which defines a type of people who are currently attending a meeting,
- The "TalkEventHost" class which defines a type of people who have invited speakers to a meeting,
- The "Speaker" class which defines a type of people are invited by another people to give a speech or presentation at a meeting
- The "Audience" class which defines a type of people who are meeting participants, but they are 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 "UniversityCampus", "Building", "Room", "OtherPlaceInBuilding" and "MeetingPlaceInBuilding". Because each of one these subclasses have specialized features, in their class definitions, they further restrict the inherited containment properties as the following:

- The "UniversityCampus" 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 the class type "Building", which means only instances of the type "Building" can be contained by an instance of this class.
- 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 "UniversityCampus", which means a building can only be a part of at most one "UniversityCampus" 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.

- 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 cannot contain any instance of the "Place" class or its subclasses.
- 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.
- 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.

Intention. The "Intention" class defines the notion of user intentions, for example, a speaker's intention to give presentation and an audience intention's to receive a copy of presentation slides and handouts. This class in defined as a union of all its subclasses, that is a collection of all defined user intentions. Because the class "SpeakerIntention is the only subclass in the current version of the ontology, the "Intention" class and the "SpeakerIntention" class have equivalent number of individuals (instances). The definition of the "SpeakerIntention" class is described in Figure 4.

4 Examples and use case

The key feature of the CoBrA ontology 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 CoBrA ontology can be used to support ontology reasoning in building a Context Broker for an intelligent meeting environment.

4.1 Use case: people presence sensors

In a pervasive computing environment, sensors are often used to detect the presence of people in a building. For example, RFID (Radio Frequency Identification) sensors can detect the presence of Smart Tags and conclude the presence of people who wear them, and Bluetooth sensors can detect the proximity presence of the Bluetooth-enabled personal devices and conclude the presence of the device owners.

Using the CoBrA ontology, these people presence sensors can effectively share people 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).

```
<cobra:Person rdf:about
    ="http://www.cs.umbc.edu/people/hchen4">
  <cobra:isCurrentlyIn rdf:resource
    ="http://www.cs.umbc.edu/ECS210I"/>
</cobra:Person>
```

Figure 5. When Harry Chen enters Room ECS210I and swipes his RFID badge at the door, the RFID sensor informs the broker of his presence in the room.

Figure 5 shows an example of the person presence information that is sent to the broker. Upon receiving this information, the broker will reason about Harry Chen's context. The following three examples describe how the broker may reason about his contexts.

Example 1: To determine if Harry Chen is in the ECS Building

A1: Person("Harry Chen") has property isCurrentlyIn("ECS210I").

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("Harry Chen") is a type of the PersonInBuilding class (i.e., Harry is currently in a building). Furthermore, because Room("ECS210I") is-PartOf the Building("ECS"), the broker can deduce Harry is currently in the ECS building.

Example 2: To determine if Harry Chen is in school today

B1: Person("Harry Chen") is in Building("ECS"). (From Example 1: A3)

B2: Building("ECS") isPartOf UniversityCampus("UMBC")

B3 <= B1+B2: Person("Harry Chen") is in school today

Example 3: To determine if Harry Chen is NOT in any rooms in the ECS building. For example, he is talking to someone in the hallway or has just left the meeting.

C1: Person("Harry Chen") is in Room("ECS210I") & Building("ECS"). (From Example 1: A3)

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

C3 <= C1+C2: It is false that Person("Harry Chen") is NOT in a room in the ECS building.

4.2 Use case: a room agent

In an intelligent space, 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 maintains 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 in the

```
<cobra:Room rdf:about
  ="http://www.cs.umbc.edu/ECS210I"/>
  <cobra:hostsMeeting rdf:resource
  ="http://www.ittalks.org/me293"/>
</cobra:Room>
```

Figure. 6 A meeting is scheduled to take place in ECS210I at 11:00am. Few minutes before the meeting, the room agent of ECS210I informs the broker that the room is about to host a meeting.

As the context of the room changes, the room agent will inform the broker of the updated contexts. Figure 6 shows an example of the information that is sent to the broker from the room agent. From this information, the broker can reason about additional context of the room and the context of people in the room. These contexts may include: 1) whether a person is currently in a meeting place, and 2) whether a person is a meeting participant of a particular meeting. The following two examples show how the broker may reason about these contexts.

Example 5: To determine if Harry Chen is currently in a meeting place in the ECS building.

E1: Person("Harry Chen") is in Room("ECS210I") and Building("ECS") (from Example 1: A3)

E2: For any room that has the property hostsMeeting() with rdfs:range limited to Meeting, the room must be a type of MeetingPlaceInBuilding (see cobra-ont.owl).

E3: Room("ECS210I") has the property hostMeeting("me239").

E4 <= E2+E3: Room("ECS210I") is a type of MeetingPlaceInBuilding

E5: 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.

E6 <= E1+E4+E5: Person("Harry Chen") is currently in a meeting place which is in the ECS building.

Example 6: To determine if Harry Chen is attending a meeting in ECS210I (i.e., is Harry Chen a meeting participant).

F1: Person("Harry Chen") is in Room("ECS210I") (From Example 1: A3)

F2: Room("ECS210I") is a type of MeetingPlaceInBuilding. (From Example 5: E4)

F3: 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 (i.e., that person is a meeting participant).

F4 <= F1+F2+F3: Person("Harry Chen") is a meeting participant.

4.3 Use case: a person agent

Person agents are specialized agents that provide personalized services for individual users [9]. In intelligent spaces, 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 for the person agent to provide these services, 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 enter the room or joins the meeting
6. The identity of people whom the person is talking to

One source from which person agents can acquire information about their users is through user behavior monitoring. For example, Harry Chen is scheduled to talk about ontology development at Wednesday's meeting. Days before the meeting, while Harry prepares his PowerPoint slides, his personal agent learns his intention to give presentation at the meeting. On the day of the meeting, as Harry enters the conference room, the personal agent informs the broker of Harry's intention and queries the broker for Harry's situational

```
<cobra:Person rdf:about
  ="http://www.cs.umbc.edu/people/hchen4">
  <cobra:hasIntention rdf:resource
    ="&cobra;#GiveSlideShowPresentation"/>
</cobra:Person>
```

Figure 7. On the day of the meeting, as Harry enters the conference room, his person agent informs the broker of his intention to give slide show presentation.

contexts. Figure 7 shows an example of the information that is sent to the broker from the person agent.

Upon 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 7: To determine the role of a person (e.g., is Harry Chen is the speaker of meeting "me239").

G1: Person("Harry Chen") is the same person as MeetingParticipant("Harry Chen") (From Example 6: F4)

G2: MeetingParticipant("Harry Chen") is associated with Meeting("me239") (From Example 5 & Example 6)

G3: Person("Harry Chen") has the intention to GiveSlideShowPresentation. (Informed by Harry Chen's person agent)

G4: If a person is a type of MeetingParticipant and that person has owl:oneOf the SpeakerIntention, then that person is **LIKELY** to be a speaker.

G5 <= G1+G2+G3: Person("Harry Chen") 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 [8]), the broker learns that some person who is a type of TalkEventHost has invited Harry Chen to the meeting "me239" (see Figure 8). This information can increase the certainty about the role of Harry Chen being a speaker.

```
<cobra:TalkEventHost rdf:about="#somePerson">
   <cobra:invites rdf:resource
     ="http://www.cs.umbc.edu/people/hchen4"/>
</cobra:TalkEventHost>
```

Figure 7. Few days before the meeting, the broker learns from a talk announcement server that some person has invited Harry Chen to give a talk.

G6: If G5 is true, and the person who in question is invited by some TalkEventHost, then that person **MUST BE** a speaker.

G7: Person("Harry Chen") is invited by a TalkEventHost.

G8 <= G6+G7: Person("Harry Chen") must be a speaker.

5 Comparisons with other work

Our work will go beyond previous work in building intelligent, context aware pervasive computing envi-

ronments and draw on recent advances in multi-agent systems, the semantic web, intelligent interfaces and wearable computing. We describe the major points of departure and the new results we will incorporate below.

Some mobile computing research projects have explored context-awareness as a means to improve the user interfaces of mobile devices (e.g., automatically initiate voice recording application when a user holds the device like a cell phone or a microphone). In our architecture, the notion of context-awareness goes beyond the basic sensing of how devices are being used and positioned. In intelligent spaces, context-awareness involves a deep understanding of situation conditions in which people, agents, services, and devices interact with each other. Previous work on context-aware architectures such as the MIT Intelligent Room [7], Georgia Tech's Context Toolkit [14], and HP's Cooltown [10] hard-coded context ontologies into the underlying system implementation. Systems developed using these architectures can not easily extended and interoperate. In our architecture, context ontologies are explicitly represented using ontology languages (i.e., RDF, OWL) allowing independently developed agents to exploit common ontologies to share knowledge and interoperate.

Although a number of agent teamwork frameworks have been developed in the past (e.g., Joint Intention [17], Joint Responsibility [18] and SharedPlans [19]), none of these theoretical frameworks is immediately applicable for building cooperate agents in a pervasive context-aware environment. We will develop a pragmatic teamwork framework to help agents to dynamically form teams, coordinate, maintain the team and eventually disband.

Previous work did not provide explicit mechanisms for detect and resolve inconsistent and ambiguous contextual knowledge. We will develop hybrid reasoning mechanisms, utilizing logic reasoning, fuzz logic, learning, and user behavior modeling, to help the broker to maintain a coherent and consistent view of the contexts of the intelligent spaces. In these and other systems, communication between distributed agents is often grounded in programming language specific APIs and ad hoc interfaces that limit agents' ability to interoperate and share knowledge. We will use and help to extend standard frameworks for agent communications and knowledge sharing, including the DARPA Grid [19] and FIPA standards [20]. Similarly, we will use semantic web languages including RDF and OWL as standards for publishing and communicating both ontologies as well as instance data.

User privacy and information security have not played a part in previous work. For example, in HP's Cooltown system and MIT Intelligent Room, agents are allowed to freely share knowledge acquired about users; in Context Toolkit, sensitive contextual information and resources are fully accessible to all agents without any restriction. In our architecture, we will develop policy-driven mechanisms to control the access and share of users' personal contexts and sensitive information.

6 Conclusion & Future Work

We have described a framework for an agent based pervasive computing environment that includes an intelligent context broker. 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.

In the next version of the CoBrA ontology, we will include additional concepts and vocabularies to model other detail aspects of meetings and potential services in the environment. In short term, we plan to prototype an ontology reasoning component for building a Context Broker. The prototype will exploit TRIPLE [21] and Jess [22].

References

[1] M. Smith, C. Welty, and D. McGuinness Web Ontology Language (OWL) Guide Version 1. 2003. Available online: http://www.w3.org/TR/owl-guide/,

[2] F. Harmelen, J. Hendler, I. Horrocks, D. McGuinness, P. Patel-Schneider, and L. Stein. Web Ontology Language (OWL) Reference Version 1.0. 2003. Available online: http://www.w3.org/TR/owl-ref/

[3] T. Berners-Lee. Primer: Getting into RDF & Semantic Web using N3. 2003. Available online: http://www.w3.org/2000/10/swap/Primer

[4] T. Berners-Lee, J. Hendler, and O. Lassila, “The Semantic Web," Scientific American, 284, 5, pp. 34-43, May. 2001.

[5] S. Peters and H. Shrobe. Using Semantic Networks for Knowledge Representation in an Intelligent Environment. Accepted for submission to IEEE International Conference on Pervasive Computing and Communications 2003 (PerCom03), Dallas-Fort Worth, Texas.

[6] H. Chen, S. Tolia, C. Sayers, T. Finin, A. Joshi, Creating Context-Aware Software Agents, GSFC/JPL Workshop on Radical Agent Concepts, NASA GSFC, Greenbelt, 26-28 Sept. 2001.

[7] M. Coen. The Future of Human-Computer Interaction or How I learned to stop worrying and love my Intelligent Room. IEEE Intelligent Systems. March/April 1999.

[8] S. Cost, T. Finin, A. Joshi, et.al., ITTALKS: A Case Study in the Semantic Web and DAML+OIL, IEEE Intelligent Systems, v17n1, Jan/Feb 2002, pp 40-47.

[9] T. Finin, A. Joshi, L. Kagal, O. Ratsimore, S. Avancha, V. Korolev, H. Chen, F. Perich, R. Cost, Information Agents for Mobile and Embedded Devices, to appear, International Journal on Cooperative Information Systems, 2003.

[10] T. Kindberg and John Barton. A Web-based nomadic computing system. Computer Networks (Amsterdam, Netherlands, 1999), 35(4):443-456, 2001.

[11] O. Lassila and D. L. McGuinness. The Role of Frame-Based Representation on the Semantic Web. Electronic Transactions on Artificial Intelligence, 2003.

[12] I. Horrocks. (2002). DAML+OIL: a Reason-able Web Ontology Language. Presented at EDBT 2002.

[13] S. Decker, S. Melnik, F. Van Harmelen, D. Fensel, M. Klein, J. Broekstra, M. Erdmann, and I. Horrocks. The Semantic Web: The roles of XML and RDF. IEEE Internet Computing. 15 (3) , pp. 63-74. 2000.

[14] G. Abowd, A. Dey, R. Orr, and J. Brotherton. Context-awareness in wearable and ubiquitous computing. In ISWC, pp. 179-180, 1997

[15] M. Weiser. The Computer for the 21st Century. *Scientific American*, 265(30):94-104, 1991

[16] O. Lassila and R. R. Swick. Resource Description Framework (RDF) Model and Syntax Specification. Available online: http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/

[17] P. Cohen, and H. J. Levesque, "Teamwork," Nous 25(4), Special Issue on Cognitive Science and Artifical Intelligence, 1991 pp. 487-512. To appear in: Handbook of MultiAgent Systems.

[18] N. R. Jennings and E. H. Mamdani: Using Joint Responsibility to Coordinate Collaborative Problem Solving in Dynamic Environments, Proc of 10th National Conf. on Artificial Intelligence (AAAI-92), San Jose, USA, 1992, 269-275.

[19] I. Foster and C. Kesselman, "The Globus Project: A Status Report", Proceedings of the Seventh Heterogeneous Computing Workshop, March 1998, IEEE Computer Society Press, pp. 4--19.

[20] FIPA abstract architecture specification. www.fipa.org

[21] M. Sintek, S. Decker: TRIPLE---A Query, Inference, and Transformation Language for the Semantic Web. International Semantic Web Conference (ISWC), Sardinia, June 2002.

[22] E. J. Friedman-Hill. Jess, The Expert System Shell for the Java Platform. Available online: http://herzberg.ca.sandia.gov/jess/docs/61/