Intelligent Agents Meet Semantic Web in a Smart Meeting Room*

Harry Chen, Filip Perich, Dipanjan Chakraborty, Tim Finin, Anupam Joshi
Department of Computer Science & Electrical Engineering
University of Maryland, Baltimore County
{hchen4, fperic1, dchakr1, finin, joshi}@csee.umbc.edu

Abstract

We describe a new smart meeting room system called EasyMeeting that explores the use of FIPA agent technologies, Semantic Web ontologies, logic reasoning, and security and privacy policies. Building on a pervasive computing system that we have developed previously, EasyMeeting can provide relevant services and information to meeting participants based on their situational needs. Our system exploits the context-aware support provided by the Context Broker Architecture (CoBrA). Central to CoBrA is an intelligent broker agent that maintains a shared model of context for all computing entities in the space and enforces the privacy policies defined by the users. We also describe the use of CoBrA ontologies, logic reasoning, and privacy protection mechanisms, and evaluate our initial user experience studies.

1 Introduction

Pervasive computing is a vision about our future life style. In this vision, computer systems will be seemlessly integrated into our everyday life, anticipating our needs and providing relvant services and information to us in an anytime any where fashion. As a step towards this vision, we have developed a smart meeting room system called **EasyMeeting** that explores the use of FIPA agent technologies, Semantic Web ontologies, logic reasoning, and security and privacy policies. In this paper, we describe the system implementation and evaluate our user experience studies.

A smart meeting room system is a distributed system that consists of communities of intelligent agents, services, devices, and sensors that share a common goal. The goal is to provide relevant services and information to the meeting participants (e.g., speakers, audiences, and organizers) based on their situational conditions (or contexts). Some typical smart meeting room applications include automatic capturing and indexing free-hand sketches on a whiteboard [12], tracking the location of and providing assistance to meeting attendees [8], teleporting graphical user interfaces [2], and assisting a group of researchers to reschedule meetings and find presenters [25].

A key to the realization of smart meeting room system is the use of context [6]. Context is any information that can be used to characterize the situation of a person or a computing entity [9]. Previous research [21, 24] has viewed location information as an important aspect of context. We believe that in addition to the location information, an understanding of context should also include information that describes system capabilities, services offered and sought, the activities and tasks in which people and computing entities are engaged, and their situational roles, beliefs, desires, and intentions.

While previous research [9, 11, 23] has successfully demonstrated the use of context in building smart spaces, we believe a great challenge remains in defining an architecture for supporting a community of context-aware agents. Some critical research issues include *context modeling*, *context reasoning*, *knowledge sharing*, and *user privacy protection* [6]. To address these issues, we have developed a broker-centric agent architecture called Context Broker Architecture (CoBrA) [6]. Key features of CoBrA include using Semantic Web languages for representing context ontologies and for supporting context reasoning, and using the Rei policy language for controlling the sharing of users' contextual information. In our EasyMeeting's user experience studies, we have successfully demonstrated the use of CoBrA for helping agents to provide context-aware services.

The rest of this document is organized as follows. Section 2 overviews EasyMeeting and its predecessor Vigil. Section 3 describes the implementation of CoBrA in supporting EasyMeeting, including CoBrA ontologies, context reasoning, and privacy protection. We evaluate our user experience studies in Section 4. Conclusions and future work are given in Section 5.

*This work was partially supported by DARPA contract F30602-97-1-0215, Hewlett Packard, NSF award 9875433, and NSF award 0209001.

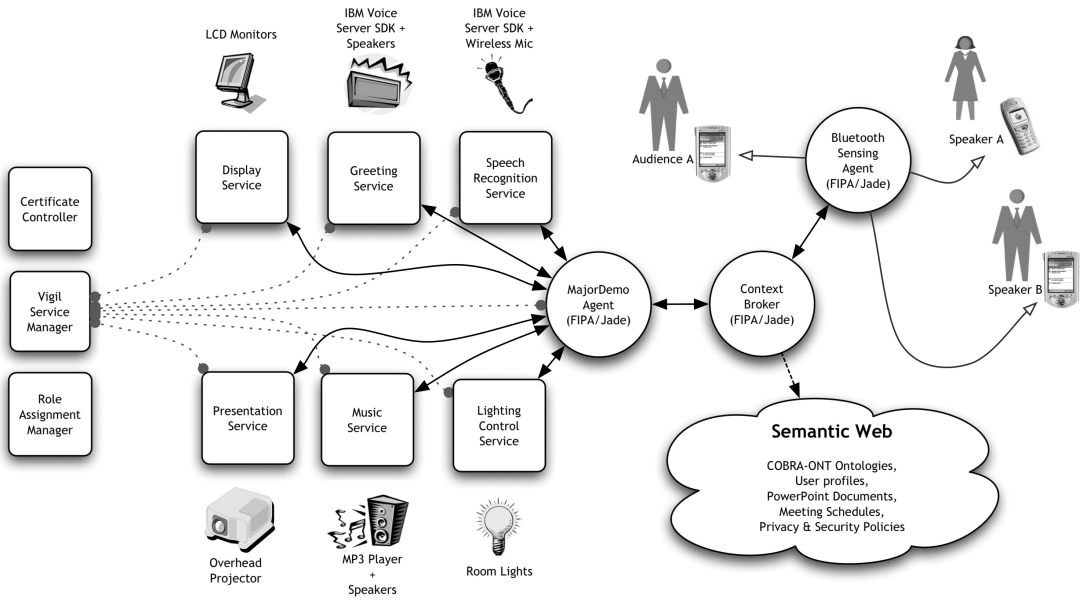

Figure 1. In EasyMeeting the Context Broker shares its contextual knowledge with the MajorDemo agent. Using this knowledge, MajorDemo selects and then invokes appropriate Vigil services to provide relevant services and information to the speakers and audiences.

2 The EasyMeeting System

EasyMeeting is an extension to Vigil [26], a third generation pervasive computing infrastructure developed at UMBC. Security is the main focuses in Vigil. While the research development behind Vigil shows great promises in building flexible and secure smart spaces [26, 7], it lacks the necessary support for context-awareness and privacy protection [6]. To improve upon the previous system, in EasyMeeting we added this support by exploiting CoBrA (see Figure 1).

2.1 Vigil Overview

The Vigil computing environment consists of clients, services, and the Vigil managers. The Vigil managers (i.e., Service Manager, Role Assignment Manager, and Certificate Controller) are specialized server entities that facilitate system communication, client role management, and service access control. The services in Vigil are computer applications that provide services to the users in a smart space (e.g., light control service, printing service, and fax service). The clients in Vigil are entities that use or receive services. A client can be a human user or a computing service.

Services usually register themselves with one or more Service Managers. The functions of a Service Manager are to provide a Yellow Page service for the clients to discover desirable services and to notify the clients of the status changes of the subscribed services (e.g., the light intensity level change in a light control service, and the playlist change in a MP3 music service).

Vigil differs from other frameworks [1, 3] in using a policy based logic inference mechanism for controlling the access to different services. When a service is registered with a Service Manager, the service defines a list of roles that can be given the permission to access its services. To prove the authenticity of this information, the Service Manager requires the list of role definition to be enclosed in a signed digital certificate. Upon receiving a digital certificate from the service, if it can be verified, the Service Manager adds the enclosed access permission information to its knowledge base.

To access a registered service, a client must first request an access permission handle (or handle for short) from the Service Manager. A handle is a digital certificate signed by a Service Manager. In order for a handle to be granted, the client must prove to the Service Manager that it fulfills one of the roles that is defined by the service. To prove a client's role, the client usually presents a role certificate that is generated by a Role Assignment Manager.

The Role Assignment Manager is a trusted server entity that can reason about the roles of a particular client based on some pre-defined system policy rules. The role based rea-

soner in this Vigil manager is implemented in Prolog based on the Rei framework [13]. A key feature of the reasoner is the use of deontic concepts (i.e., rights, prohibitions, obligations, and dispensations) to construct logic inference rules for reasoning about client roles. For example, a client can delegate its right to other clients to access a particular services, a Service Manager can revoke a client's right to access some restricted services, and a service can prohibit certain clients from accessing its services.

2.2 EasyMeeting Services

The goal of developing EasyMeeting is to create a smart meeting room that can facilitate typical user activities in an everyday meeting. Our current implementation provides context-aware support for helping speakers and audiences during a meeting presentation. In the future, we will expand our context-aware support to include services for tracking the location of an absent meeting participant, tracking the availability of a portable projector device that is shared within a department, and exchanging contact information between the visitors and other meeting attendees. We have developed the following six services:

1. **Speech Recognition Service** This service can be invoked to perform speech recognition on a set of predefined voice input vocabularies (e.g., "yes", "no", "show Harry's presentation") and generate CCML (Centaurus Capability Markup Language) [14] commands for controlling other Vigil services. In this service, the underlying voice recognition procedure is implemented using the IBM WebSphere Voice Server SDK and Voice XML.

2. **Presentation Service** This service can be invoked to display PowerPoint presentations on an overhead project in the room. The presentation file is fetched from a URL that a client has specified. It defines a set of CCML commands for controlling the flow of a displaying presentation (e.g., "next", "back", "stop", "start"). Exploiting the Speech Recognition Service, users can control their presentations via speech commands.

3. **Lighting Control Service** This service can be invoked to adjust the lighting conditions in a meeting room. The underlying lighting control mechanism is implemented using the X10 technology. Lights are wired to X10 Lamp Modules. Using the light control API, this service can control the lights to be turned on or off, or to be dimmer or brighter.

4. **Music Service** This service can be invoked to play audio music files that are accessible on the Web using an existing MP3 music player software. A typical use of this service is to play background music while participants are waiting for a meeting to begin.

5. **Greeting Service** This service can be invoked to play a specified greeting message. The played audio file is dynamically generated from a client specified greeting message (e.g., "Welcome to the eBiquity Group, President Hrabowski"). The underlying text-to-speech procedure is implemented using the IBM WebSphere Voice Server SDK and Voice XML.

6. **Display Service** This service can be invoked to instruct all subscribed web browsers to display any URL (e.g., some user's home page). The intended use of this service for displaying speaker profiles or reference material on the handheld devices that individual audiences carry. This is useful for helping audiences to learn about the background of the speaker or to obtain references about the material that is currently presented.

2.3 Context-Awareness in EasyMeeting

Exploiting the notion of meeting context is a key feature in EasyMeeting. The purpose of acquiring meeting context is to help the computer system to decide what services and information it should provide to the meeting participants based on their situational needs. Without necessarily requiring the participants to give explicit instructions, using context can help to facilitate their meeting-related tasks.

In EasyMeeting, the role of a Context Broker is provide a shared model of context for all agents and services. In particular, it is responsible for acquiring and maintaining consistent knowledge about (i) the location of meeting participants, (ii) the event schedule of a meeting, (iii) the presentations that are scheduled for the meeting, (iv) the profiles of the presentation speakers, and (v) the state of a meeting. To acquire this knowledge, the Context Broker explores different sources of information that is published on the Semantic Web and that is provided by the sensor agents (e.g., the Bluetooth Sensing Agent).

The role of a MajorDemo agent is to decide when and what services should be provided to the meeting participants. It relies on the Context Broker to provide information about the meeting context and uses the registered Vigil services to facilitate different meeting related tasks. In order to simultaneously interact with the Vigil services and the Context Broker, this agent is implemented with a hybrid design that bridges the API's for invoking services in Vigil and for communicating with the agents in CoBrA.

The following is a typical EasyMeeting use case: Room 338 is a smart meeting room. On January 8th, 2004, a presentation is scheduled to take place from 1:00-2:30 PM in this room. Moments before the event starts, the room's

Context Broker acquires the meeting's schedule from the Semantic Web and concludes the meeting is about to take place in the Room 338. As the meeting participants begin to arrive, the room's Bluetooth Sensing Agent detects the presences of different Bluetooth enabled devices (e.g., cellphones, PDA's). Since each device has a unique device profile that is represented using standard device ontologies, the sensing agent can share this information with the Context Broker.

Based on the user profile information stored in the knowledge base of the Context Broker (e.g., who owns what devices), the Context Broker concludes the owners of the detected devices are also located in the Room 338. In the group of the arrived participants, Harry (the speaker) and President Hrabowski (the distinguished guest) are two people that are listed in the meeting schedule. The Context Broker shares the location information of these listed participants with the subscribed MajorDemo agent.

Knowing that President Hrabowski is a distinguished guest, the MajorDemo agent invokes the Greeting Service to greet him. At 1:00 PM, the Context Broker informs the MajorDemo agent that all listed *key* participants have arrived and that the presentation can start. Knowing all the lights in the meeting are currently switched on and the background music is also playing, the agent invokes the Dim Light Method on the the Light Control Service and the Stop Music Method on the Music Service.

As Harry walks to the front of the meeting room, he speaks to the system using a wireless microphone, "load Harry's presentation". The voice command is received by the Voice Recognition Service and a corresponding CCML command is generated. The MajorDemo agent sends this text string command to the Presentation Service along with the URL at which Harry's presentation can be downloaded (this information is provided by the Context Broker). As the Presentation Service loads Harry's PowerPoint slides, the MajorDemo agent invokes the Display Service to show Harry's home page. A few seconds later, all LCD displays sitting on the conference table start showing Harry's biosketch and his profile. Using the same wireless microphone, Harry speaks to the system to control his presentation.

3 Context Broker Architecture

CoBrA is a broker-centric agent architecture for supporting context-aware systems in smart spaces. Central to the architecture is the presence of a Context Broker, an intelligent agent that runs on a resource-rich stationary computer in the space. In addition to its responsible for acquiring and maintaining context knowledge, it is also responsible for reasoning about the information that cannot be directly acquired from sensors (e.g., intentions, roles, temporal and spatial relations), detecting and resolving inconsistent knowledge that is stored in the shared model of context, and protecting user privacy by enforcing policies.

All computing entities in a smart space are presumed to have prior knowledge about the presence of a context broker, and all agents are presumed to communicate with the Context Broker using the standard FIPA Agent Communication Language and the ontologies defined by CoBrA.

The Context Broker consists of the following components:

- **Context Knowledge Base:** a persistent storage of the context knowledge. All context knowledge is represented in RDF triples (i.e., subject, property, object). RDF triples are stored in a persistent relational database (i.e., MySQL) backed by the Jena 2 Semantic Web Framework [4]. The stored knowledge can be accessed by other components using the standard Jena API's.

- **Context Reasoning Engine:** a rule-based inference engine for reasoning over the stored context knowledge. It defines the implementation for (i) deducing context knowledge through ontology inference (using the inference facility provided by the Jena framework) and (ii) deducing knowledge using domain heuristic rules (using Jess a CLIPS-like rule engine in Java). We will discuss the implementation of this component in detail in Section 3.2.

- **Context Acquisition Module**: a library of procedures for acquiring contextual information from different sources. The goal is to create a middle-ware abstraction to hide the low-level complexity in context acquisition (e.g., sensing, information discovery, and data mining). This approach is similar to the role of the Context Widgets in the Context Toolkit framework [9]. We have developed procedures for fetching ontology documents, user profiles, and meeting schedules from the Semantic Web, and for sensing the presence of Bluetooth enabled devices in a room.

- **Policy Management Module**: inference rules for deciding whether a particular agent has the right to access certain type of contextual information about a user. The role of this component can be viewed as the "conscience" of a Context Broker. Before the Context Broker shares an user's information with another agent, it uses this component to determine the rights for the other agent to receive this knowledge and shares the information only if the user defined policy allows. At present, this component has not yet been incorporated into EasyMeeting. In Section 3.3 we will describe our future implementation in detail.

3.1 Semantic Web Ontologies in CoBrA

CoBrA differs from other frameworks in using Semantic Web languages for representing context ontologies and for supporting context reasoning [5]. By using Semantic Web languages, CoBrA can help independently developed agents to share context knowledge and thus minimizing the cost of context sensing. Additionally, ontologies represented in the Semantic Web languages also provide a means for agents to reason about context knowledge, e.g., using a set of standard ontology axioms [19].

CoBrA ontology (or COBRA-ONT) is defined using the Web Ontology Language OWL, the latest Semantic Web language standard recommended by W3C. The current EasyMeeting implementation uses the version 0.4 of COBRA-ONT[1], which defines ontologies for action, agent, device, meeting, digital documents, space and time. When defining our ontology, some ontological constructs are adopted from the existing consensus ontologies (i.e., Friend-Of-A-Friend (FOAF), DAML-Time and the entry sub-ontology of time [18], the spatial ontologies in OpenCyc [16], Region Connection Calculus (RCC) [22], and the FIPA device ontologies[10]). Due to space limitation, in this document we only describe the key COBRA-ONT concepts that are used in the EasyMeeting.

Describing User Profiles

The user profile of a person can provide useful information for reasoning about the person's role and intentional actions, and the person's relation to certain devices. A typical user profile consists of information that describes (i) the background information of a person (e.g., contact information, employment information, professional associations, and preferences), (ii) the person's social relations to other people (e.g., whom my friends are, and whom I work with), (iii) the profiles of the person's mobile devices and personal agents (e.g., the type of communication interfaces and the display resolutions that a device supports, the agent ID of a personal agent), (iv) the person's daily meeting schedule and associated roles and preferences in the scheduled events (e.g., the time and location of a meeting that the person plans to attend, the slides of a presentation that the person often uses).

In EasyMeeting, the user profile of a person is publicly accessible on the person's home page. Let's consider part of Harry Chen's user profile, which is available at `http://www.umbc.edu/~hchen4/aboutMe`. This user profile is an OWL ontology document (i.e., a set of RDF statements) that describes information using COBRA-ONT. In the profile, an RDF resource URI `http://umbc.edu/~hchen4` is used to represent the person Harry Chen. To describe his role, this URI is defined as an individual of the `aca:GradStudentResearcher` class, which is a subclass of the class type `aca:Researcher` and `agt:Person`. The RDF resource URI that represents Harry Chen (i.e., the subject in an RDF statement) has a number of defined properties. They include the property `agt:ownsDevice` that defines a mobile device that Harry owns (e.g., a SonyEricsson T68i cellphone), the property `agt:intends` that defines an action that Harry intends to perform (e.g., an action is `ParticipateMeeting` and its object is a scheduled meeting event that Harry is known to be a participant of), and the property `aca:oftenUsedSlides` that defines a PowerPoint document that Harry often uses when giving his presentation.

[1]COBRA-ONT is available at `http://cobra.umbc.edu`

Describing Meeting Schedules

Knowing the schedule information of a meeting can help a Context Broker to reason about a meeting's context. This includes the temporal state of the meeting event, the attendance of the meeting participants, and the roles of the participants.

Similar to the user profile of a person, the schedule information of a meeting is also defined using COBRA-ONT and can be made publicly accessible via the Web (e.g., `http://cobra1.cs.umbc.edu/meetings/meeting-11212003`). A typical meeting schedule ontology document contains information about (i) the type of the meeting that has been scheduled (e.g., a demonstration session, a group meeting, or a video conference meeting), (ii) the location at which the meeting will be held, (iii) the begin and the end time of the meeting, and (iv) a list of expected participants of the meeting.

Let's consider the defined schedule information for the meeting *meeting-11212003*. In this ontology document, the RDF resource URI `http://cobra1.cs.umbc.edu/meetings/meeting-11212003#aDemoSession` represents the meeting event that is in consideration. It is an individual of the `demo:Demo` class, indicating the meeting is a type of demonstration session. This demonstration session has three demonstrators (i.e., Harry Chen, Filip Perich, and Dipanjan Chakraborty) and two distinguished audiences (i.e., President Hrabowski and Dr. Carmi). The URI that represent each of the demonstrators and the audiences are defined as an individual of the `demo:Demonstrator` and `demo:DistinguishedAudience`, respectively. Additionally, the URI that represents the meeting has a number of properties, which include the property `mto:location` that defines the meeting location, and the property `tme:hasIntervalDescription` that defines the time interval that represents the begin and the end time of the meeting.

Describing Time and Space

Ontologies of time and space are extremely useful for reasoning about context [5]. For example, the time ontologies can help a Context Broker to reason about the temporal orders among different events in a meeting room, and the space ontologies can help a Context Broker to reason about the location context of a person.

The basic representation of time is a *time entity*, which can be either an *instant* or an *interval*. An instant (or a point of time) is described by the typical calendar/clock concepts (e.g., second, minutes, hour, day, month, year, and time zone). An interval is described by two different instants, the beginning and the ending of the interval. For representing relations between instants and intervals, COBRA-ONT defines temporal relation properties, such as before, after, inside, and during. In EasyMeeting, some typical uses of the time ontologies include describing the time at which a Bluetooth enabled device is detected, the arrival time of a person, and the interval during which a meeting takes place.

The present space ontology in COBRA-ONT focuses on the symbolic representation of space, as oppose to the geospatial representation of space (e.g., GPS coordinates and GIS descriptions). The basic representation of space is a *space entity*. Space can be divided according to its geographical attributes (e.g., geo-political region, geo-political entity, country, state, and city). For modeling the environment that surrounds a meeting room, COBRA-ONT defines concepts for representing university campus, building, and room.

A simple model for describing spatial relations is the *part-whole* relation model, in which spatial entities are related to each other by either `spatiallySubsumedBy` or `spatiallySubsumes` property. To model more complex spatial relations, COBRA-ONT defines ontology constructs based on the Region Connection Calculus (RCC).

Some typical uses of the space ontology include describing the spatial model of a smart meeting room (e.g., RM 338 is a part of the ITE building, which is located on the UMBC campus, and UMBC is located in Baltimore, Maryland, USA), the location of a meeting, and the location of a person or a device.

3.2 Context Reasoning in CoBrA

The reasoning in the Context Broker exploits the OWL ontology axioms and logic inference rules. In the current implementation, the ontology reasoning is backed by the Jena rule engine and its Java API. To reason about the contextual information that cannot be inferred using ontology axioms only, the Context Broker uses a forward-chaining inference procedure defined in Jess (Java Expert System Shell).

Context Reasoning Algorithm

Rules defined in Jess are executed as part of the Context Broker's reasoning implementation. The following is a high-level description of the reasoning algorithm: When a piece of contextual information is asserted into the knowledge base, the Context Broker first selects the type of context it attempts to infer (e.g., the location of a person or the state of a meeting). If such information is unknown, the Context Broker decides whether such type of context can be inferred using only ontology reasoning. If logic inference is required, the Context Broker attempts to find all essential supporting facts by querying the ontology model. After collecting all supporting facts, the Context Broker converts the RDF representation of the facts into the corresponding Jess representation and asserts them into the Jess engine. After executing the pre-defined forward-chaining procedure, if any new facts can be deduced, the Context Broker adds their corresponding RDF representation into the ontology model.

In EasyMeeting, the logic inference procedure helps a Context Broker to reason about the state of a meeting (i.e., pre-meeting, meeting in session, post-meeting), the arrival of an anticipated meeting participant based on the presence of their personal devices, and the absence of an anticipated meeting participant.

Assumption-Based Reasoning

We recognize the implementation of the current logic inference procedure is rigid. We are investigating an assumption-based reasoning approach [17] to improve the reasoning flexibility. We plan to use the *Theorist* framework developed Poole [20], which is a Prolog meta-interpreter for processing assumption-based reasoning. Different from the conventional deductive reasoning systems, in this framework, the premises of the logic inference consists both facts (axioms given as true) and assumptions (instances of the possible hypotheses that can be assumed if they are consistent with the facts). Supporting both default reasoning and abductive reasoning is a key feature of the *Theorist* framework [20].

One way to use *Theorist* in CoBrA is for context reasoning, exploiting both default and abductive reasoning. In this approach, all contextual information acquired by the Context Broker are viewed as its observation about the environment. When an observation is received, the Context Broker first uses abduction to determine the possible causes and then uses default reasoning to predict what else will follow from the causes [17].

Let's consider the following example:

```
H1: locatedIn(Per,Rm), owner(Per,Dev)
    => locatedIn(Dev,Rm).
H2: locatedIn(Per,Rm), meeting(Mt,Rm),
    speakerOf(Per,Mt),
```

```
    not(notLocatedIn(Per,Rm))
    => intends(Per,give_prst(Mt)).

F1: locatedIn(t68i,rm338).
F2: owner(harry,t68i).
F3: meeting(m1203,rm338).
F4: speakerOf(harry,m1203).
```

Hypotheses `H1` states that a personal device is located in a room if the owner of the device is also in that room. Hypotheses `H2` states that if a person is in a room where a meeting is scheduled to take place, the same person is the speaker of the meeting, and no evidence showing the person is not in that room, then the person intends to give a presentation at the meeting. Fact `F1` states that Cellphone T68i is located in the room RM338. Fact `F2`, `F3`, and `F4` state that Harry is the owner of the Cellphone T68i, Meeting m1203 is scheduled to take place in the room RM338, and Harry is the speaker of the Meeting m1203, respectively. We expect `F1` to be knowledge acquired from the sensors, and `F2`, `F3`, and `F4` to be knowledge acquired from the Semantic Web.

Our first objective is to infer the cause for the observation that the Cellphone T68i is located in the room RM338 (i.e., `F1`). We use abduction. Based on the given knowledge, {`locatedIn(harry,rm338)`, `owner(harry,t68i)`} is a plausible explanation for `locatedIn(t68i,rm338)`. Knowing Harry is in the room RM338, our second objective is to predict his intention in that room. We use default reasoning. Using `H2`, we can infer Harry intends to give a presentation in the Meeting m1203.

3.3 Privacy Protection in CoBrA

CoBrA follows the principle of proximity and locality [15] in designing its privacy protection mechanism. The idea is to exploit the locality information of the users when enforcing access restrictions to their personal information (e.g., for different types of smart spaces, the Context Broker enforces different restrictions for others to access users' private information). CoBrA allows users to adjust levels of privacy protection using privacy policies (or policies).

Policies is a set of declarative rules that a user defines to restrict the access to his/her personal information. CoBrA uses the Rei policy language [13] to define privacy policies. Rei is a policy language for modeling rights, prohibitions, obligations, and dispensations in the domain of security. In addition to its rich support for modeling security objects, the RDF representation of the Rei language also allows for greater interoperability between CoBrA and other Semantic Web components.

A typical use case of Rei is that a user grants or prohibits some agent the right to access his/her contextual information. For example, Harry grants the MajorDemo agent the right to access his location context if the agent also is in that room, which can be expressed as

```
has(majorDemo,
right(accessContext(harry,location),
colocated(harry,majorDemo))).
```

Sometimes users may desire the Context Broker to reveal certain high-level description of their context (e.g., okay to tell agents that I'm in UMBC) but to hide only the low-level details of their context (e.g., not okay to tell agents that I'm in Room 338). We plan to develop such type of spatial granularity reasoning in the future version of CoBrA.

4 Evaluation

To evaluate the feasibility of EasyMeeting, we have conducted three different user experience studies. In each one of the studies, we invite people who are outside our research team to participate. The participants include UMBC university administrators, and visitors from commercial companies and other universities.

Each of our studies consists of a live demonstration of the features of a smart meeting room. The demonstration is similar to the use case described in Section 2.3. Before the demonstration, we set up distinctive user profiles for each participant and post these ontology documents on the Web. Each participant is also given a Bluetooth enabled mobile device and asked to active the Bluetooth connection when arrives at the meeting room. Additionally we post an ontology document that describe the schedule of the meeting on the Web. During the demonstration session, the visitors and speakers are greeted by the smart meeting room when their Bluetooth devices are detected by the sensor agent. After all participants have arrived, the smart meeting room dims the room lights and turns off the background music that is playing. The speakers can use speech to ask the system to load presentations (described in their profiles) that have been pre-fetched by the smart meeting room.

In general, the feedback from all the participated users were positive. Most of the users were very pleased and excited about a meeting room that could speak and perform actions on people's behalf. However, some users showed great concerns for their personal privacy and information security. Some user believed that in order for the general population to adopt smart meeting room systems, it is necessary to provide users with the necessary tools to control the sharing and the use of their private information.

5 Conclusions

Working towards the vision of pervasive computing, we believe that the realization of smart spaces (e.g., smart meeting rooms) requires a new approach that integrates the

use of FIPA agent technologies, Semantic Web ontologies, logic reasoning, and security and privacy policies. We have successfully developed and demonstrated EasyMeeting, a smart meeting room system that builds on CoBrA and Vigil. Our initial studies show user privacy protection is of great concern for the future success of smart spaces and pervasive computing.

We are working with other researchers to define a standard ontology for supporting pervasive computing applications called **SOUPA** – Standard Ontology for Ubiquitous and Pervasive Applications[2]. In addition, part of our short term objective is to implement privacy protection mechanisms for EasyMeeting using Rei and to enhance the logic reasoning of the Context Broker using the *Theorist* framework. Our long term objective is to enhance the context-awareness of EasyMeeting by deploying a wider variety of sensors and intelligent agents and to study their effects on user privacy and information security.

References

[1] J. Al-Muhtadi, R. Campbell, A. Kapadia, D. Mickunas, and S. Yi. Routing through the mist: Privacy preserving communication in ubiquitous computing environments. In *Proceedings of the International Conference of Distributed Computing Systems (ICDCS 2002)*, Vienna, Austria, 2002.

[2] 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.

[3] R. Campbell, J. Al-Muhtadi, P. Naldurg, G. Sampemane1, and M. D. Mickunas. Towards security and privacy for pervasive computing. In *Proceedings of International Symposium on Software Security*, Tokyo, Japan, 2002.

[4] J. Carroll, I. Dickinson, C. Dollin, D. Reynolds, A. Seaborne, and K. Wilkinson. Jena: Implementing the semantic web recommendations. Technical Report HPL-2003-146, Hewlett Packard Laboratories, 2003.

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

[6] H. Chen, T. Finin, and A. Joshi. Semantic web in in the context broker architecture. In *Proceedings of PerCom 2004*, March 2004.

[7] A. Choudhri, L. Kagal, A. Joshi, T. Finin, and Y. Yesha. PatientService : Electronic Patient Record Redaction and Delivery in Pervasive Environments. In *Fifth International Workshop on Enterprise Networking and Computing in Healthcare Industry (Healthcom 2003)*, June 2003.

[8] M. H. Coen. Design principles for intelligent environments. In *Proceedings of AAAI/IAAI 1998*, pages 547–554, 1998.

[9] A. K. Dey. *Providing Architectural Support for Building Context-Aware Applications*. PhD thesis, Georgia Institute of Technology, 2000.

[10] Foundation for Intelligent Physical Agent. *FIPA Device Ontology Specification*, pc00091a edition, 2001.

[11] 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.

[12] T. Hammond, K. Gajos, R. Davis, and H. Shrobe. An agent-based system for capturing and indexing software design meetings. In *Proceedings of the International Workshop on Agents in Design (WAID'02)*, Cambridge, MA, August 2002.

[13] L. Kagal, T. Finin, and A. Joshi. A Policy Based Approach to Security for the Semantic Web. In *2nd International Semantic Web Conference (ISWC2003)*, September 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*, 2001.

[15] M. Langheinrich. Privacy by design–principles of privacy-aware ubiquitous systems. In *Proceedings of UbiComp 2001: International Conference on Ubiquitous Computing*, 2001.

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

[17] A. K. MacKworth, R. G. Goebel, and D. I. Poole. *Computational Intelligence: A Logical Approach*, chapter 9, pages 319–342. Oxford University Press, 1998.

[18] F. Pan and J. R. Hobbs. Time in owl-s. In *Proceedings of AAAI-04 Spring Symposium on Semantic Web Services*, Stanford University, California, 2004.

[19] P. F. Patel-Schneider, P. Hayes, and I. Horrocks. OWL web ontology language semantics and abstract syntax. 2003.

[20] D. Poole. Compiling a default reasoning system into prolog. *New Generation Computing*, 9(1):3–38, 1991.

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

[22] D. A. Randell, Z. Cui, and A. G. Cohn. A spatial logic based on regions and connection. In *Proceedings of the 3rd International Conference on Knowledge Representation and Reasoning*, 1992.

[23] M. Roman and R. H. Campbell. Gaia: Enabling active spaces. *9th ACM SIGOPS European Workshop*, September 2000.

[24] A. Roy, S. K. D. Bhaumik, A. Bhattacharya, K. Basu, D. J. Cook, and S. K. Das. Location aware resource management in smart homes. In *First IEEE International Conference on Pervasive Computing and Communications (PerCom'03)*, 2003.

[25] M. Tambe, P. Scerri, and D. V. Pynadath. Adjustable autonomy for the real world. In *Proceedings of AAAI Spring Symposium on Safe Learning Agents 2002*, 2002.

[26] J. Undercoffer, F. Perich, A. Cedilnik, L. Kagal, A. Joshi, and T. Finin. A secure infrastructure for service discovery and management in pervasive computing. *The Journal of Special Issues on Mobility of Systems, Users, Data and Computing*, 2003.

[2] See also `http://pervasive.semanticweb.org`