# A Context Broker for Building Smart Meeting Rooms

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

**Abstract**

Building smart meeting rooms requires the support of a computing system architecture. In this paper, we describe the Context Broker Architecture (CoBrA), a broker-centric agent architecture for pervasive context-aware systems. CoBrA exploits the Web Ontology Language OWL for supporting knowledge sharing and data fusion, uses logic inferences for resolving and detecting inconsistent context knowledge, and provides users with a policy language to control their private information. Central to CoBrA is an intelligent broker agent that maintains a share model of context for all agents, services, and devices in the space, and protects users' privacy by enforcing the policy rules that they have defined. We also describe the use of CoBrA ontologies, context reasoning mechanisms, and privacy protection in a smart meeting room system called EasyMeeting.

## Introduction

In the pervasive computing vision, computer systems are seamlessly integrated into the life of everyday users, providing services to users in an anywhere-and-any-time fashion. A key aspect of the future computing environment is context-awareness, which can be defined as a computer system's ability to provide relevant services and information to users based their situational conditions.

Building context-aware systems for an open and dynamic environment, e.g., smart meeting rooms, intelligent homes, smart vehicles, can be difficult and costly without the adequate support of a computing architecture (Chen & Kotz 2000; Chen, Finin, & Joshi 2003). Key research challenges include defining an explicit representation of context that is suitable for knowledge sharing and data fusion, constructing reasoning mechanisms for detecting and resolving inconsistent contextual knowledge, and implementing an adequate framework for user privacy protection.

To address these issues, we propose a new system architecture called the Context Broker Architecture (CoBrA). CoBrA differs from other similar architectures (Salber, Dey, & Abowd 1999; Schilit 1995; Coen 1998; Peters & Shrobe 2003) in exploiting the Web Ontology Language OWL to define ontologies for supporting knowledge sharing and data fusion, using logic inferences for detecting and resolving inconsistent context knowledge that is acquired from unreliable physical sensors, and using the Rei policy language (Kagal, Finin, & Joshi 2003) to give users the control of their contextual information.

The rest of this document is organized as the following: First, we describe the shortcomings of previous pervasive context-aware architectures. Second, we present the design and implementation of CoBrA. Third, we describe two smart meeting room applications that we plan to implement for demonstrating the feasibility of CoBrA. Lastly, we state our future works and conclusions.

## Background

Context is any information that can be used to characterize the situation of a person or a computing entity (Dey 2000). While others have viewed location as an important aspect of context (Lamming & Flynn 1994; Priyantha, Chakraborty, & Balakrishnan 2000; Kindberg & Barton 2001; Lin, Laddaga, & Naito 2002), in addition to which we believe an understanding of context should also include information that describe system capabilities, service offered and sought, the activities in which people and computing entities are engaged, the spatial and temporal properties associated with the tasks that the users perform, and their situational roles, beliefs, desires, and intentions.

A number of pervasive computing architectures have been developed in the past. Research in building the previous architectures have made progress in various aspects of pervasive computing (e.g., developed a middle-aware framework to facilitate context acquisition (Dey 2000), defined new extensible programming libraries for building intelligent room agents (Coen *et al.* 1999), and created badge-size tracking devices for determining people's location in an indoor environment (Schilit 1995; Want *et al.* 1992)).

Major shortcomings of these systems are weak in supporting knowledge sharing and reasoning and lack of adequate user privacy protection. In the Context Toolkit framework (Dey 2000), Schilit's context-aware architecture (Schilit 1995), and the Active Badge system (Want *et al.* 1992), context knowledge is embedded in programming objects (e.g. Java classes) that are often inadequate for supporting knowledge sharing and data fusion operations. The designs of these systems also make strong assumptions about

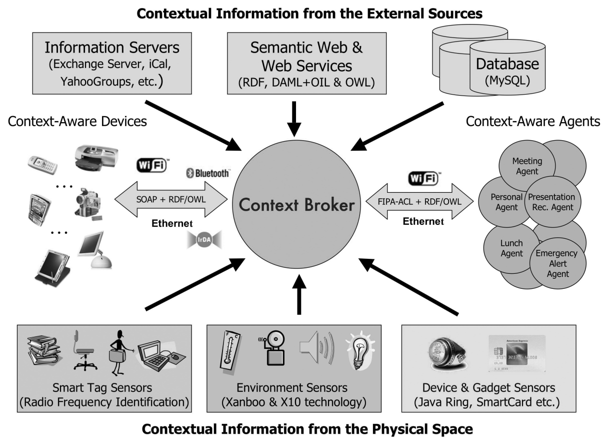
Contextual Information from the External Sources
Information Servers (Exchange Server, iCal, YahooGroups, etc.)
Semantic Web & Web Services (RDF, DAML+OIL & OWL)
Database (MySQL)
Context-Aware Devices
Context-Aware Agents
SOAP + RDF/OWL
Ethernet
Context Broker
FIPA-ACL + RDF/OWL
Ethernet
Meeting Agent
Personal Agent
Presentation Rec. Agent
Lunch Agent
Emergency Alert Agent
Smart Tag Sensors (Radio Frequency Identification)
Environment Sensors (Xanboo & X10 technology)
Device & Gadget Sensors (Java Ring, SmartCard etc.)
Contextual Information from the Physical Space

Figure 1: A Context Broker acquires contextual information from heterogeneous sources and fuses into a coherent model that is then shared with computing entities in the space.

the accuracy of the information acquired from the hardware sensors. In an open and dynamic environment, such assumptions can lead to system implementations that cannot cope with the frequently occurred inconsistent context knowledge. In the Intelligent Room system (Coen 1998) and the Cooltown architecture (Kindberg & Barton 2001), information about a user can be freely shared by all computing entities in the environment. As the physical environments are populated with ambient sensors, users may be unaware of the use and the sharing of their private information, which can create great concerns for privacy.

## Context Broker Architecture

Central to CoBrA is an intelligent agent called *Context Broker* (see Figure 1). The role of this agent is to maintain a model of the present context and to share this model of context knowledge with other agents, services and devices. A smart space environment may be populated with multiple Context Brokers, and each broker is responsible to maintain parts of the space's context. For example, in a smart space that encloses a university building, different Context Brokers may be designated to maintain the context of different classrooms, conference rooms, hallways, elevator, etc. As different Context Brokers may possess distinctive context knowledge, agents can subscribe to the Context Brokers and acquire different context knowledge and fuse them to form a coherent view of the smart space context.

The key components of a Context Broker:

- *CoBrA Ontology (COBRA-ONT)*: a set of ontologies for agents to describe contextual information and to share context knowledge.
- *Context Reasoning Engine (CoRE)*: a logic inference engine for reasoning with ontologies, for interpreting context using acquired situational information, and for detecting and resolving inconsistent context knowledge.
- *Module for Privacy Protection (MoPP)*: a policy language for users to define privacy protection rules and an inference engine for deciding the permission to share a user's contextual information.

## CoBrA Ontology (COBRA-ONT)

We believe a key requirement for realizing context-aware systems is the use of ontologies. COBRA-ONT is an ontology for supporting pervasive context-aware systems (Chen, Finin, & Joshi 2003). This ontology, expressed in the Web Ontology Language OWL, defines concepts for representing actions, agents, devices, meetings, time, and space (see Figure 2).

The OWL language is a Semantic Web language standard backed by the W3C. As oppose to the use of other knowledge representation scheme (e.g., semantic networks (Peters & Shrobe 2003)), we believe the OWL language is more suitable for expressing information that is to be exchanged and shared by distributed computing entities. It has rich ex-

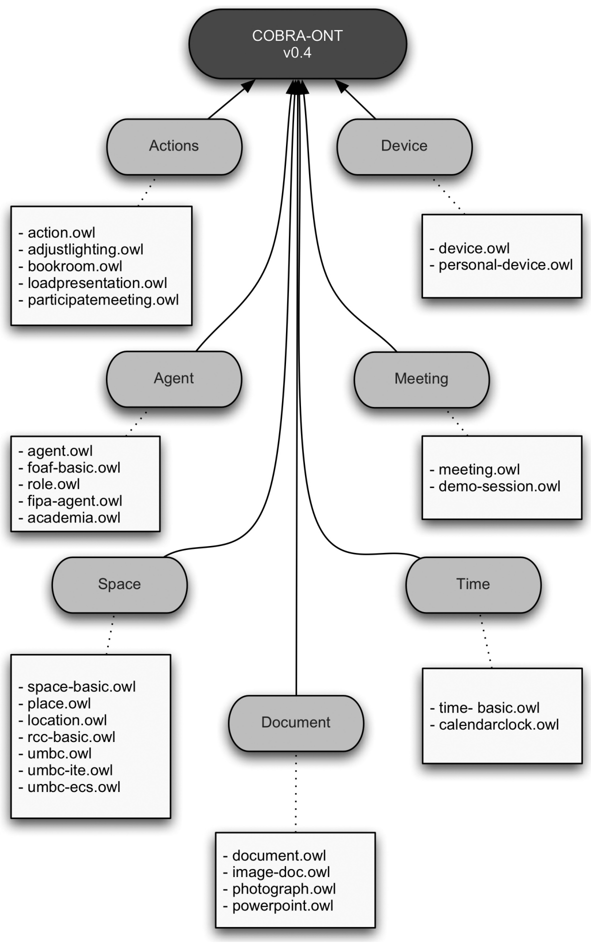

Figure 2: The structure layout of COBRA-ONT v0.4. Ontologies are expressed using the OWL-DL subset of the OWL language. COBRA-ONT is available at `http://cobra.umbc.edu/`

pressive power for defining complex ontologies, and it has standard language syntaxes (e.g., XML, N3, N-Triple) for computer programs to process and manipulate represented information. Furthermore, because the OWL language can also be used as a meta-language to define other special purpose languages, e.g., security policy language (Kagal, Finin, & Joshi 2003), agent communication language (Zou *et al.* 2003), by defining the ontologies using OWL can increase the interoperability between different system components.

To avoid defining ontologies completely from scratch, we choose to adopt ontology terms and organizations from other consensus ontologies, which includes the DAML-Time/Time-Entry ontology (Hobbs 2002), the OpenCyc spatial ontologies, the Friends-Of-A-Friend (FOAF) ontology[1], and the FIPA device ontology[2]. The rationale behind our approach is to avoid importing a substantial amount of irrelevant foreign ontologies into the CoRE. For example, the OpenCyc ontology that is published on the DAML.org web site consists of more than 200,000 statements. Directly importing all these ontologies could hinder the performance of the existing ontology reasoning system. In the future, when the implementation of the ontology reasoning system matures, we plan to rely on the OWL ontology mapping mechanism (Smith, Welty, & McGuinness 2003) to support reasoning with the foreign ontologies.

To illustrate the use of COBRA-ONT, we describe an example ontology and show how a Context Broker can use this ontology to reason about context. In this example, our goal is to develop a smart meeting room system for the Information Technology and Engineering building (ITE) on the UMBC campus. For the Context Broker that is designated to maintain the context of a conference room (e.g., Room ITE-201A), one of its tasks is to monitor the location of different devices and people.

Let's assume a Bluetooth device sensor in the Room ITE-201A detects the present of a SonyEricsson T68i cellphone at 14:30:00 on Dec. 1, 2003. To notify the Context Broker of this information, the sensor composes a description of this event using COBRA-ONT (see Figure 3). The URI `http://umbc.edu/~hchen4/myT68i` represents the SonyEricsson T68i cellphone that the sensor has detected, and the URI `http://umbc.edu/ITE/RM-201A` represents the Room ITE-201A.

As the Context Broker receives this notification, it can infer certain properties about the device that the sensor has detected. According to the device ontologies defined in COBRA-ONT, all individual of the class `SonyEricssonT68i` are also type of the class `Device-SupportsBluetooth` and the class `Cellphone`. Knowing this information, the Context Broker can infer the device, identified by the URI `http://umbc.edu/~hchen4/myT68i` is a cellphone that support Bluetooth connectivity.

Knowing the location at which the cellphone is detected, the Context Broker can infer additional location context of the device using ontologies. Based on the UMBC spatial ontology (see Figure 4), the device's location context includes other geographical regions that spatially contains the Room ITE-201A, and they are the ITE building, the UMBC campus, the Baltimore County, the Maryland state, and the US. To derive this conclusion, the inference relies on two property constructs that are predefined in the spatial ontology of COBRA-ONT: (i) all spatial relations between different geographical regions are sub-properties of the `inRegion` property, (ii) the `inRegion` property is a type of the OWL transitive property.

[1] `http://xmlns.com/foaf/0.1/`

[2] `http://www.fipa.org/specs/fipa00091/XC00091C.pdf`

```
<loc:LocationAtTimeInstant rdf:about="urn:event3231">
  <loc:object>
    <dev:SonyEricssonT68i
       rdf:about="http://umbc.edu/~hchen4/myT68i">
      <spc:objectFoundInLocation
         rdf:about="http://umbc.edu/ITE/RM-201A"/>
    </dev:SonyEricssonT68i>
  </loc:object>

  <tme:hasInstantDescription>
    <tme:InstantDescription>
      <tme:definedByCalendar>
         <calc:CalendarDescription>
            <calc:year>2003</calc:year>
            <calc:month>12</calc:month>
            <calc:dateOfMonth>1</calc:dateOfMonth>
        </calc:CalendarDescription>
      </tme:definedByCalendar>

      <tme:definedByClock>
         <calc:ClockDescription>
            <calc:hourOfDay>14</calc:hourOfDay>
            <calc:minute>30</calc:minute>
            <calc:second>00</calc:second>
         </calc:ClockDescription>
      </tme:definedByClock>
    </tme:InstantDescription>
  </tme:hasInstantDescription>
</loc:LocationAtTimeInstant>
```

Figure 3: An ontology that describes the presence of a Sony-Ericsson T68i cellphone that has been detected in the Room 201A on Dec. 1, 2003 at 14:30:00.

## Context Reasoning Engine (CoRE)

In addition to ontology inference, Context Brokers can also use logic inference to reason about contextual information. The role of CoRE is to provide the Context Broker the ability to interpret certain types of contextual information that otherwise cannot be easily deduced using only ontology inferences. While the underlying RDF data model of the OWL language is suitable for reasoning about the semantic relations between the physical objects and abstract concepts, but it has limited built-in support for other types of logic inferences, for example, it does not support default reasoning and uncertainty reasoning.

CoRE has a two-tiers design. Both tiers share a common knowledge base, which stores information acquired from the external sources and the knowledge that is deduced from the

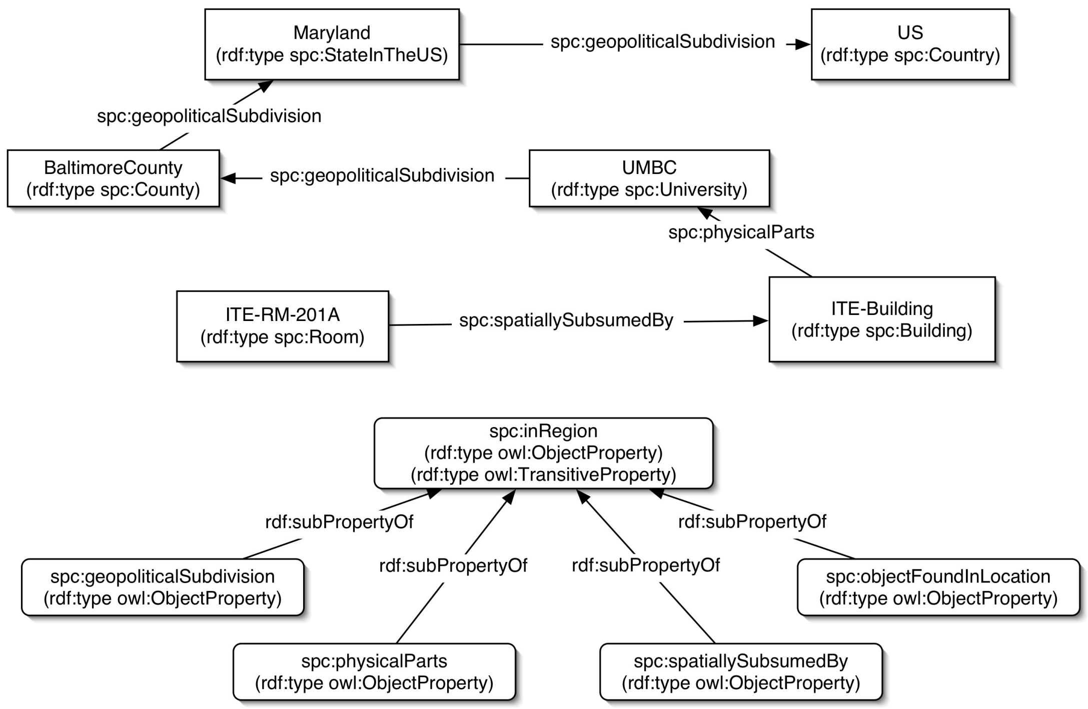
Maryland
(rdf:type spc:StateInTheUS)
spc:geopoliticalSubdivision
US
(rdf:type spc:Country)
spc:geopoliticalSubdivision
BaltimoreCounty
(rdf:type spc:County)
spc:geopoliticalSubdivision
UMBC
(rdf:type spc:University)
spc:physicalParts
ITE-RM-201A
(rdf:type spc:Room)
spc:spatiallySubsumedBy
ITE-Building
(rdf:type spc:Building)
spc:inRegion
(rdf:type owl:ObjectProperty)
(rdf:type owl:TransitiveProperty)
rdf:subPropertyOf
rdf:subPropertyOf
rdf:subPropertyOf
rdf:subPropertyOf
spc:geopoliticalSubdivision
(rdf:type owl:ObjectProperty)
spc:objectFoundInLocation
(rdf:type owl:ObjectProperty)
spc:physicalParts
(rdf:type owl:ObjectProperty)
spc:spatiallySubsumedBy
(rdf:type owl:ObjectProperty)

Figure 4: A simple spatial ontology about UMBC is defined using COBRA-ONT. This ontology enables the Context Broker to reason about the location context of a device or a person. Ontologies are shared between the Context Broker and other computing entities in the space.

logic inferences. In Tier-1, a set of inference rules is defined to reason over contextual information using the ontologies defined in OWL and COBRA-ONT. Inference results are asserted into the knowledge base. In Tier-2, a different set of inference rules is defined to reason over contextual information using domain heuristics. Inference results are also asserted into the knowledge base.

In our CoRE prototype implementation, the two-tiers design is realized using the Jena 2 semantic web framework (`http://jena.sourceforge.net/`) and the XSB system (Sagonas *et al.* 2003). The knowledge base shared between the Tier-1 and Tier-2 is implemented as a Persistent Ontology Model in Jena, which is a RDF data storage backed by a relational database (e.g., MySQL). The ontology inference in Tier-1 uses the OWL reasoner provided by the Jena 2 framework, and the inference rules in Tier-2 are implemented as Prolog modules in the XSB system.

At present, the ontology inferences in the Tier-1 are limited to the OWL-lite subset of the OWL language. An example use of the ontology inference is to reason about the location context of a device (see the example described in the previous section).

In Tier-2, our prototype implementation consists of two XSB modules: (i) rules for temporal reasoning based on the DAML-Time axioms and Allen's temporal interval calculus (Allen 1983), (ii) rules for interpreting the location context of a person and the status of a meeting.

Our temporal reasoning rules are built on two basic temporal concepts: time instant and time interval. Time instants are point-like that they have no interior points, and time intervals are things with extent and points (Hobbs 2002). A time instant is represented as a Prolog function `t_instant/1`. A time interval is represented as a Prolog function `t_interval/2` whose first and second argument represent the beginning time instant and the ending time instant of an interval, respectively. Figure 5 shows the implementation of some basic temporal reasoning rules. In addition to the listed rules, we have also implemented a set of temporal reasoning rules based on Allen's temporal interval calculus, such as equals, meets, during, overlaps, finished by, etc.

The second XSB module defines a set of rules for reasoning about (i) the correlation between the location of a person and the personal devices that the person owns; (ii) the state of a meeting (i.e., pre-meeting, meeting-in-session, or post-meeting) based on the location of the key meeting participates (i.e., speakers, organizers).

To determine if a person is located in a particular room at a particular time instant, we define rules to reason if any personal devices (e.g., cellphones) that the person owns is located in the room at that particular time instant, and there is no evidence showing that the person is located in some other room. Using the time interval inference rules, the correlation between the location of a person and his/her devices can also be inferred if the temporal events are represented in

```
% begins(+TimeInstant1,+TimeInstant2).
begins(t_instant(T), t_instant(T)).

% begins(?TimeInstant,+TimeInterval)
begins(t_instant(T1), t_interval(t_instant(T2),_)) :-
   begins(t_instant(T1), t_instant(T2)).

% ends(+TimeInstant1,+TimeInstant2).
ends(t_instant(T), t_instant(T)).

% ends(?TimeInstant,+TimeInterval)
ends(t_instant(T1), t_interval(_,t_instant(T2))) :-
   ends(t_instant(T1), t_instant(T2)).

% inside(+TimeInstant,+TimeInterval)
inside(t_instant(T1),
       t_interval(t_instant(BT), t_instant(ET))) :-
   before(t_instant(BT),t_instant(T1)),
   after(t_instant(ET),t_instant(T1)).

% begins_or_in(+TimeInstant,+TimeInterval)
begins_or_in(t_instant(T),
         t_interval(t_instant(BT),t_instant(ET))) :-
   begins(t_instant(T),
      t_interval(t_instant(BT),t_instant(ET)));
   inside(t_instant(T),
      t_interval(t_instant(BT),t_instant(ET))).

% time_between(+TimeInterval,+TimeInstant,+TimeInstant)
time_between(t_interval(t_instant(BT),t_instant(ET)),
                        t_instant(A),t_instant(B)) :-
   begins(t_instant(A),
          t_interval(t_instant(BT),t_instant(ET))),
   ends(t_instant(B),
        t_interval(t_instant(BT),t_instant(ET))).
```

Figure 5: An example of the temporal reasoning rules that are defined in the CoRE. These rules can be used to interpret certain contextual information that otherwise cannot be easily inferred using only ontology reasoning.

time intervals or a mix of time instants and intervals.

To determine the state of a meeting, we define rules with the following heuristics:

- The state of a meeting is *pre-meeting* (i.e., a scheduled meeting has not yet begun) if the present time instant is *inside* the time interval of the meeting schedule, and all key meeting participants *are not currently located in* the meeting room.
- The state of a meeting is *meeting-is-session* (i.e., a scheduled meeting is currently happening) if the present time instant is *inside* the time interval of the meeting schedule, and all key meeting participants *are currently located in* the meeting room.
- The state of a meeting is *post-meeting* (i.e., a scheduled meeting has ended) if the scheduled end time of the scheduled meeting is *before* the present time instant, and all key meeting participants *are not currently located in* the meeting room.

We recognize that the logic inferences in our present implementation are rigid, e.g., the rules do not address the condition in which key participates are temporarily absent from the meeting or the condition in which the end time of a schedule meeting has passed, but the key participates of the meeting remain in the room. In the future, we plan to explore the use of an assumption-based reasoning framework developed by Poole (Poole 1991) as a means to improve inference flexibility. The use of this framework is described later in the Future Works section.

## Module for Privacy Protection (MoPP)

To protect user privacy, the design of MoPP follows the principle of proximity and locality (Langheinrich 2001), exploiting the locality information of the users when enforcing access restrictions to their private information. MoPP allows different sets of access control model to be “plugged into” the Context Broker for different types of the smart spaces. An access control model consists of a set of inference rules that a Context Broker uses to decide what access restrictions should be imposed on the sharing of a particular type of user private information.

To override the default access model, users can inform the Context Broker of their own privacy policy. Upon receiving the user policy, through MoPP the Context Broker reasons the access restrictions entailed from the policy and compares which with the default policy. If it differs from the default policy but without conflict, then the user defined restrictions take the precedence. If conflict exists, the Context Broker will inform the user of the conflicts, inquiring the user for manual resolutions.

The privacy language in CoBrA is defined based on the Rei policy language, which consists of an ontology for modeling rights, prohibitions, obligations and dispensations. Ontologies in Rei are expressed in the RDF language (Kagal, Finin, & Joshi 2003). The first two examples in Figure 6 show how the Rei policy language can be used to defined privacy policies.

Protecting user privacy sometimes means to hide the details of a user’s contextual information. To allow users to have a fine-grained control over their contextual information, our privacy policy language extends the Rei language to include granularity control parameters. Example 3 in Figure 6 shows how to define a policy rule with a granularity control parameter. In this example, the effect of the rule is to instruct the Context Broker to keep secrete about all Alice’s location information except for information that describes a geographical region whose physical coverage area has radius larger than 1 mile. In other words, assuming Alice is located in the Room ITE-201A, if some agent asks if Alice is located on the UMBC campus, the Context Broker will reply “yes”, and if some agent asks if Alice is located in the ITE building, the Context Broker will reply “unknown”.

A key issue in building a privacy protection infrastructure is *the problem of inference*. In an open and dynamic environment, sometimes a user’s private information can be inferred from his/her public information. Here is some typical examples that involve the problem of inference: (i) if someone knows the home phone number of a user, it is possible to acquire the mailing address of the user by looking up a

**Example 1**: A projector device has the right to access the location context of the user Alice only if the device is located in the same room as Alice.

```
has(projector, right(accessContext(alice, location), colocated(alice, projector)))
```

**Example 2**: A location tracking service is prohibited from access the location context of the user Alice when the service is not authenticated by the context broker.

```
has(locTracker, prohibition(accessContext(alice, location), not(authBy(locTracker, broker))))
```

**Example 3**: Alice does not permit others to know the specific room that she is in, but she permits others to know the general area in which she is located.

```
has(broker, right(shareContext(alice, location))), granularity(location, raduis(1, mile)).
```

Figure 6: Examples of user-defined privacy policy rules, expressed in the Prolog syntax of the Rei policy language. Privacy policy rules are processed by the MoPP to decide the appropriate restrictions that should be imposed when sharing the user's contextual information.

white-page service, (ii) if someone knows the email address of a user (e.g., `alice@whitehouse.gov`), based on the domain name part of the address, it is possible to infer the profile of the user (e.g., the owner of the previous email address is affiliated with the US White House).

To address the problem of inference, we propose a meta-reasoning approach to detect the potential leakage of private information by analyzing user defined policies. In this approach, MoPP is equipped with a set of rules that defines the potential inferences from one type of information to another type. The following is some examples of the rules:

- Some agent $X$ may be capable of inferring the location a user $Y$ if $X$ knows the daily schedule of $Y$.

```
mayKnow(X,location(Y) :- know(X,schedule(Y)).
```

- Some agent $X$ may be capable of inferring the home address of a user $Y$ if $X$ knows the phone number of $Y$.

```
mayKnow(X,homeAdd(Y)) :- know(X,phoneNum(Y)).
```

Based on these rules, the meta-reasoning engine in MoPP can help the Context Broker to decide if additional rules should be included into the privacy policy of a user in the case in which certain publicly accessible information can be used to derive the user's private information. If the Context Broker decide additional policy rules should be included, it will notify the user and inquire the user's decision.

## EasyMeeting Applications

To demonstrate the feasibility of CoBrA, we plan to prototype a smart meeting room system called **EasyMeeting**. Using CoBrA as the system foundation, EasyMeeting aims to provide services and relevant information to meeting participants based on their situational needs. In the rest of this section, we describe two services that make uses of the Context Brokers in a smart space.

### Intelligent Personal Agent

An Intelligent Personal Agent (or personal agent for short) is a software agent that maintains personal information for a user. It usually operates on a stationary computer that the user has previously set up (e.g., on a desktop computer in the office or at home). A personal agent can access the user's private information, such as the daily schedules, address books, personal profiles, location information, etc. It also has the right to decide when and with whom this information can be shared.

Key functions of the personal agent are to keep track of a user's context (e.g., what the user is doing, where the user is located, what event the user is attending) and to share this information with other agents that attempt to provide context-aware services for the user. A typical use case of the personal agent is the following:

As the user Alice enters a smart meeting room ITE-201A, the Context Broker of the associated space immediately informs Alice's personal agent. Knowing Alice is located in ITE-201A, the Context Broker attempts to determine why Alice is there. It asks Alice's personal agent. After reviewing Alice's daily schedule, the personal agent discovers that Alice is scheduled to give a talk in ITE-201A. Knowing Alice's is the speaker of the meeting, and she is about to give a presentation, the personal agent informs the Context Broker of Alice's role at the meeting and the URL of the PowerPoint slides that Alice will be using. On receiving this information, the Context Broker shares it with a projector service agent, which has been priorly granted permission to acquire Alice's contextual information. Few minutes later, the projector service agent downloads the slides and sets up the presentation.

### Projector Tracking Service

A Projector Tracking Service is a service that monitors the whereabouts of a portable projector. This service aims to reduce the amount of human efforts required to administrate the usage of a public projector. A public projector is a device that is owned by an organization (e.g., a department) but shared by different people in the organization (e.g., faculties, graduate students).

Key functions of this service include providing updated location information about a projector and sending reminders to the user who has previously borrowed the projector but has not yet return the device. A typical use case of this service is the following:

As Alice starts to give her PowerPoint presentation, the Context Broker detects the presence of a projector in the room. Immediately the Context Broker informs the Projector Tracking Service of the projector location.

Knowing the projector is in the Room ITE-201A, the service asks the Context Broker to provide a contact person who is responsible for returning the device when the presentation ends. From the meeting schedule, the Context Broker learns that Alice is invited by Bob, who is the organizer of the meeting. Knowing this information about Bob, the Context Broker sends a SMS message to Bob's cellphone, inquiring if he is willing to be in charge of the returning of the projector. Bob replies "yes". The Context Broker sends Bob's contact information to the Project Tracking Service. To keep track of the projector's location, the Projector Tracking Service subscribes to the Context Broker, requesting to be notified about the device location and the state of the device (i.e., active, sleep, turned off) every 30 minutes.

As the meeting ends, Bob leaves the room and forgets to return the projector. Couple hours later, the Projector Tracking Service continuously receives updates about the projector being inactive in the Room ITE-201A. Without having any evidence to the contrary, the service concludes Bob has forgotten to return the device. Immediately, it sends a reminder to Bob asking him to return the projector.

## Future Work

The short term objective of our project is to improve the logic inference mechanism in CoRE, to define the privacy policy language using OWL, and to prototype meta-reasoning engine to support the previously described privacy protection mechanism in MoPP. Our long-term objective is to prototype the EasyMeeting system, using which to demonstrate and evaluate the feasibility of CoBrA.

In order to improve the logic inference in CoRE, we are investigating the use of the *Theorist* framework (Poole 1991), a Prolog meta-interpreter for processing assumption-based reasoning. Differ 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.

One way to use *Theorist* 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 (MacKworth, Goebel, & Poole 1998).

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.

## Conclusions

We believe building smart spaces such as smart meeting rooms requires a broker-centric agent architecture that uses ontologies to support knowledge sharing and data fusion, uses logic inferences to resolve and detect inconsistent context knowledge, and supports policy-based privacy protection. The design of CoBrA is a new approach to help distributed agents, services, and devices to share context knowledge and protect user privacy.

Based our preliminary work in using the OWL language, we believe it is suitable for defining ontologies for modeling contextual information, supporting context reasoning, and facilitating knowledge sharing in a pervasive context-aware system. In the course of prototyping our system, we found the Jena 2 semantic web framework to be useful for manipulating and processing Semantic Web ontologies and for building simple inference engines. We believe as the Semantic Web tools and other related technologies emerges, they will create new research opportunities for building pervasive context-aware systems.

## Acknowledgments

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

## References

Allen, J. F. 1983. Maintaining knowledge about temporal intervals. *Communications of the ACM* 26(11):832–843.

Chen, G., and Kotz, D. 2000. A survey of context-aware mobile computing research. Technical Report TR2000-381, Dartmouth College, Computer Science, Hanover, NH.

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

Coen, M.; Phillips, B.; Warshawsky, N.; Weisman, L.; Peters, S.; and Finin, P. 1999. Meeting the computational needs of intelligent environments: The metaglue system. In *Proceedings of In 1st International Workshop on Managing Interactions in Smart Environments (MANSE'99)*.

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

Dey, A. K. 2000. *Providing Architectural Support for Building Context-Aware Applications*. Ph.D. Dissertation, Georgia Institute of Technology.

Hobbs, J. R. 2002. A daml ontology of time. `http://www.cs.rochester.edu/~ferguson/daml/daml-time-20020830.txt`.

Kagal, L.; Finin, T.; and Joshi, A. 2003. A policy language for a pervasive computing environment. In *IEEE 4th International Workshop on Policies for Distributed Systems and Networks*.

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

Lamming, M., and Flynn, M. 1994. Forget-me-not: intimate computing in support of human memory. In *Proceedings FRIEND21 Symposium on Next Generation Human Interfaces*.

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

Lin, J.; Laddaga, R.; and Naito, H. 2002. Personal location agent for communicating entities (PLACE). In *4th International Symposium on Human Computer Interaction with Mobile Devices*.

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

Peters, S., and Shrobe, H. 2003. Using semantic networks for knowledge representation in an intelligent environment. In *1st Annual IEEE International Conference on Pervasive Computing and Proceedings of the 1st Annual IEEE International Conference on Pervasive Computing and Communications (PerCom'03)*.

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

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

Sagonas, K.; Swift, T.; Warren, D. S.; Freire, J.; Rao, P.; Cui, B.; and Johnson, E. 2003. *The XSB Programmers' Manual*, version 2.6 edition.

Salber, D.; Dey, A. K.; and Abowd, G. D. 1999. The context toolkit: Aiding the development of context-enabled applications. In *Proceedings of CHI'99*, 434–441.

Schilit, W. N. 1995. *A System Architecture for Context-Aware Mobile Computing*. Ph.D. Dissertation, Columbia University.

Smith, M. K.; Welty, C.; and McGuinness, D. 2003. Owl web ontology language guide. `http://www.w3.org/TR/owl-guide/`.

Want, R.; Hopper, A.; Falcao, V.; and Gibbons, J. 1992. The active badge location system. Technical Report 92.1, Olivetti Research Ltd., ORL, 24a Trumpington Street, Cambridge CB2 1QA.

Zou, Y.; Finin, T.; Ding, L.; Chen, H.; and Pan, R. 2003. Using semantic web technology in multi-agent systems: a case study in the taga trading agent environment. In *Proceeding of the 5th International Conference on Electronic Commerce*.