Semantic Web in a Pervasive Context-Aware Architecture*

Harry Chen
University of Maryland
Baltimore County
hchen4@cs.umbc.edu

Tim Finin
University of Maryland
Baltimore County
finin@cs.umbc.edu

Anupam Joshi
University of Maryland
Baltimore County
joshi@cs.umbc.edu

ABSTRACT

This document describes a new approach that explores the use of Semantic Web languages in building an architecture for supporting context-aware systems. This new architecture called Context Broker Architecture (CoBrA) differs from other architectures in using the Web Ontlogy Language OWL for modeling ontologies of context and for supporting context reasoning. Central to our architecture is a 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 and its associated ontologies in prototyping an intelligent meeting room.

1. INTRODUCTION

The Semantic Web, described by Tim Berners-Lee *et. al.*[4], is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation . A key difference between the Semantic Web and the present Web lies in the representation of information. In the present Web, the representation is meant for machines to process information at the syntax level. In the future Semantic Web, however, the representation allows machines to process and reason about information at the semantic level. In this paper, we describe a new approach that explores the use of Semantic Web technologies (i.e., languages, logic inferences, and programming tools) in building an architecture for supporting context-aware systems in smart spaces (e.g., intelligent meeting rooms, smart vehicles, and smart houses).

Context is any information that can be used to characterize the situation of a person or a computing entity [8]. Previous research [16, 23] have viewed location information as an important aspect of context. We believe 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.

The dynamic nature of a smart space environment creates great challenges for developing context-aware systems. We believe some of the critical research issues are context modeling, context reasoning, knowledge sharing, and user privacy protection. To address these issues, we propose an agent-oriented architecture called Context Broker Architecture that exploits Semantic Web languages to model ontologies of context, reason with context in a smart space, and define a policy language for users to control their context information.

The rest of this document is organized as the following: Section 2 gives a brief overview of the Semantic Web and the Web Ontology Language OWL. In Section 3 we describe the rationale behind our Semantic Web approach to building a new architecture for context-aware systems. Section 4 presents the design of CoBrA and its use case scenario in an intelligent meeting room. Section 5 describes our preliminary work on prototyping **EasyMeeting**, an intelligent meeting room system that builds on the design of CoBrA. Discussions of the related work and concluding remarks are given in Section 6 and Section 7, respectively.

2. AN OVERVIEW OF THE SEMANTIC WEB

The Semantic Web is a vision in which web pages are augmented with information and data that is expressed in a way that facilitates its understanding by computing machines [1]. The current human-centered web is largely encoded in HTML, which focuses largely on how text and images would be rendered for human viewing. Over the past few years we have seen a rapid increase in the use of XML as an alternative encoding, one that is intended primarily for machine processing. The machine which process XML documents can be the end consumers of the information or they can be used to transform the information into a form appropriate for human understand (e.g., as HTML, graphics, and synthesized speech). As a representation language, XML provides essentially a mechanism to declare and use simple data structures and thus leave much to be desired as a language in which to express complex knowledge. Enhancements to basic XML, such XML Scheme, address some of the shortcomings, but still do not result in an adequate language for representing and reasoning about the kind of knowledge essential to realizing the Semantic Web vision.

A goal of the Semantic Web initiatives sponsored by the World Wide Web Consortium (W3C) is to develop languages that are adequate for representing and reasoning about the semantics of information on the Web. The Web Ontology Language OWL is the latest standard proposed by the Web-Ontology Working Group. The OWL language builds on XML's ability to define customized taggging schemes and RDF's flexible approach to representing data [15]. Due to space limitation, in this section we describe some of the OWL language constructs and show how they are used to

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

```
<rdf:RDF
  xmlns="http://example.org/devices#"
  xmlns:exd="http://example.org/devices#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#">

<owl:Ontology rdf:about="http://example.org/devie">
  <rdfs:comment>
    An ontology about people and devices.
  </rdfs:comment>
  <rdfs:label>An Example Ontology</rdfs:label>
</owl:Ontology>

<owl:Class rdf:ID="Person"/>
<owl:Class rdf:ID="Device"/>

<owl:Class rdf:ID="Cellphone"/>
  <rdfs:subClassOf rdf:resource="#Device"/>
</owl:Class>

<owl:Class rdf:ID="PDA"/>
  <rdfs:subClassOf rdf:resource="#Device"/>
</owl:Class>

<owl:DatatypeProperty rdf:ID="name">
  <rdfs:domain
    = "http://www.w3.org/2002/07/owl#Thing"/>
  <rdfs:range rdf:resource
    = "http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>

<owl:ObjectProperty rdf:ID="ownedBy">
  <rdfs:domain rdf:resource="#Device"/>
  <rdfs:range rdf:resource="#Person"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="owns">
  <owl:inverseOf rdf:resource="#ownedBy"/>
</owl:ObjectProperty>

<Person rdf:ID="P1">
  <name>"Harry Chen"</name>
  <owns rdf:resource="#D1"/>
</Person>

<Person rdf:ID="P2">
  <name>"Joe Smith"</name>
</Person>

<Cellphone rdf:ID="D1">
  <name>"Harry's Blue Phone"</name>
</Cellphone>

<PDA rdf:ID="D2">
  <name>"Blue Knight"</name>
  <ownedBy rdf:resource="P2"/>
</PDA>

<rdf:RDF>
```

Figure 1: An example of OWL classes and properties of a simple ontology.

define ontologies.

OWL is a language for defining and instantiating ontologies. An ontology is a formal explicit description of concepts in a domain of discourse (or classes), properties of each class describing various features and attributes of the class, and restrictions on properties [20]. The normative OWL exchange syntax is RDF/XML. OWL ontologies are usually placed on web servers as web documents, which can be referenced by other ontologies and downloaded by applications that use ontologies. Figure 1 shows an example of an OWL ontology encoded in RDF/XML.

The definition in our example ontology has two parts. The first part is a set of classes and properties that describe people, devices, and the ownership relation between them. The second part is a set of class individuals that represent specific people and devices in some imaginary domain discourse.

The beginning of our ontology defines a set of XML namespaces declarations, enclosed in an opening `rdf:RDF` tag, that indicate what specific vocabularies are being used. The first two declarations identify the namespace associated with this ontology. The first makes it the default namespace, stating that unprefixed names refer to our example ontology. The second identifies the namespace of our example ontology with the prefix `exd:`.

The third namespace declaration says that in this document, elements prefixed with `owl:` should be understood as referring to things drawn from the OWL namespace (`http://www.w3.org/2002/07/owl#`). This declaration introduces the OWL vocabulary.

OWL depends on constructs defined by RDF, RDF-S, and XML Schema datatypes. In our example, the `rdf:` prefix refers to things drawn from the namespace called `http://www.w3.org/1999/02/22-rdf-syntax-ns#`. The next two namespace declarations make similar statements about the RDF Schema (`rdfs:`) and XML Schema datatype (`xsd:`) namespaces.

After the namespace declarations, a collection of assertions about the ontology is grouped under the `owl:Ontology` tag. These assertations define the meta-data for the example ontology document. The `rdfs:comment` tag encloses a short comment about this document. The `rdfs:label` tag encloses a string that is meant to be a human-readable version of the ontology label.

The class definition in our example begins with the concept `Person` and `Device`. A class represents a set of individuals in the domain (i.e., `Person` represents a set of individual person, `Device` represents a set of individual device). `PDA` and `Cellphone` are two other classes that represent a set of individual PDA and cellphones in our domain. They are both subclasses of the class `Device`. Note that, by default, every individual in the OWL world is a member of the class `owl:Thing`. Thus, all of our defined classes are implicitly subclasses of `owl:Thing`.

Properties in OWL lets us assert general facts about the members of classes and specific facts about individuals [22]. A property is a binary relation. Two types of properties are distinguished: datatype properties and object properties. The former is relations between instances of classes and RDF literals and XML Schema datatypes. The latter is relations between instances of two classes.

When defining a property, its domain and range can be specified. Specifying the domain of a property asserts that the domain value of the property must belong to a specified class. Specifying the range of a property asserts that the range value of the property must belong to a specified class or to a specified data type.

In our example, the `name` property is defined as a type of the datatype property. It has domain `owl:Thing` and range `http://www.w3.org/2001/XMLSchema#string`. This asserts that any individual member of the

owl:Thing class can have a name property with some string value. For example, the individual P1 is instantiated with the name stirng "Harry Chen", and individual D1 is instantiated with the name string "Harry's Blue Phone".

There are two object properties in our ontologies. First, the ownedBy property has domain Device and range Person. This asserts that the ownedBy property can relate a device to its owner (e.g., the device D2 is owned by the person P2). The owns property is defined as an inverse of the ownedBy property. This asserts that for every triple (X, ownedBy, Y), there is a triple (Y, owns, X) and vice versa (since the inverse relation is symmetric). Thus, from the example, since we know (P1, owns, D1), we can conclude (D1, ownedBy, P1), and similiarly since we know (D2, ownedBy, P2), we can conclude (P2, owns, D2) also holds.

3. WHY SEMANTIC WEB

A key requirement for realizing context-aware systems is to give computer systems the ability to understand their situational conditions. To achieve this, it requires contextual information to be represented in ways that are adequate for machine processing and reasoning. We believe the Semantic Web languages are well suited for this purpose for the following reasons:

- Ontologies expressed in the Semantic Web languages provide a means for independently developed context-aware systems to share context knowledge, minimizing the cost of and redundancy in sensing.
- RDF and OWL are knowledge representation languages with rich expressive power that are adequate for modeling various types of contextual information, e.g., information associated with people, events, devices, places, time, and space.
- Because context ontologies have explicit representations of semantics, they can be reasoned by the available logic inference engines. Systems with the ability to reason about context can detect and resolve inconsistent context knowledge that often result from imperfect sensing.
- The Semantic Web languages can be used as meta-languages to define other special purpose languages such as communication languages for knowledge sharing, policy languages for privacy and security [12]. A key advantage of this approach is better interoperability. Tools for langages that share a common root of constructs can better interoperate than tools for languages that have diverse roots of constructs.

4. CONTEXT BROKER ARCHITECTURE

The use of Semantic Web languages and tools are key features in our Context Broker Architecture. In CoBrA, the OWL language is used to define ontologies for modeling context, providing common vocabaries for knowledge sharing, reasoning about the relation between the context and the domain heuristic knowledge, and expressing user defined policies. The core of our architecture is a specialized server entity called *context broker*.

In a smart space a context broker has the following responsibilities: (i) provide a centralized model of context that can be shared by all devices, services, and agents in the space, (ii) acquire contextual information from sources that are unreachable by the resource-limited devices, (iii) reason about contextual information that cannot be directly acquired from the sensors (e.g., intentions, roles, temporal and spatial relations), (iv) detect and resolve inconsistent knowledge that is stored in the shared model of context, and (v) protect user privacy by enforcing policies that the users have defined to control the sharing and the use of their contextual information.

4.1 An Intelligent Meeting Room Scenario

The design of CoBrA is aimed to support context-aware systems in smart spaces. The following is a typical use case scenario of CoBrA in an intelligent meeting room system:

R210 is an intelligent meeting room with RFID sensors[1] embedded in the walls and furniture for detecting the presence of the users' devices and clothing. As Alice enters the room, these sensors inform the R210 broker that a cell phone belonging to her is present and the broker adds this fact in its knowledge base.

As she sits, the agent on Alice's Bluetooth enabled cell phone discovers R210's broker and engages in a "hand shake" protocol (e.g. authenticates agent identities and establishes trust [12]) after which it informs the broker of Alice's privacy policy. This policy represents Alice's desires about what the broker should do and includes (i) the context information about Alice that the broker is permitted or prohibited from storing and using (e.g., yes to her location and roles, no to the phone numbers she calls), (ii) other agents that the broker should inform about changes in her contextual information (e.g., keeping Alice's personal agent at home informed about her location contexts), and (iii) the permissions for other agents to access Alice's context information (e.g., all agents in the meeting room can access Alice's contexts while she is in the room).

After receiving Alice's privacy policy, the broker creates a profile for Alice that defines rules and constraints the broker will follow when handling any context knowledge related to Alice. For example, given the above policy, the profile for Alice would direct the broker (i) to acquire and reason about Alice's location and activity contexts, (ii) to inform Alice's personal agent at home when Alice's contexts change, and (iii) to share her contexts with agents in the meeting room.

Knowing Alice's cell phone is currently in R210 and having no evidence to the contrary, the broker concludes Alice is also there. Additionally, because R210 is a part of the Engineering building, which in turn is a part of the UMBC campus, the broker concludes Alice is located in Engineering building and on campus. These conclusions are asserted into broker's knowledge base.

Following the rules defined in the profile, the broker informs Alice's personal agent of her whereabouts. On receiving this

[1]RFID stands for Radio Frequency Identification (see also http://www.rfid.org)

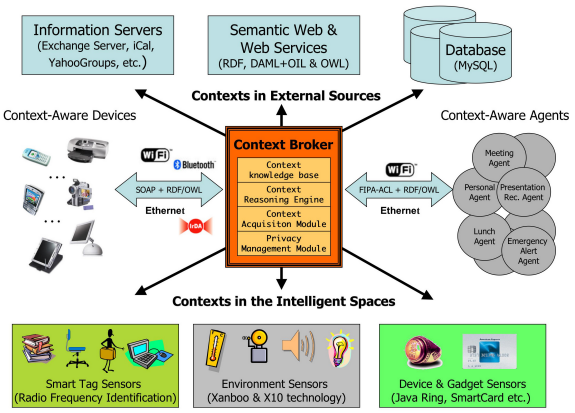

Figure 2: A context broker acquires contextual information from heterogeneous sources and fuses it into a coherent model that is then shared with computing entities in the space.

information about Alice, her personal agent attempts to determine why Alice is there. Her Outlook calendar has an entry indicating that she is to give a presentation on the UMBC campus about now, so the personal agent concludes that Alice is in R210 to give her talk and informs the R210 broker of it's belief.

On receiving information about Alice's intention, the R210 broker shares this information with the projector agent and the lighting control agent in the ECS 210. Few minutes later, the projector agent downloads the slides from Alice's personal agent and sets up the projector, the lighting control agent dims the room lights.

4.2 Context Broker

Figure 2 shows the design of a context broker. In smart spaces, context brokers are assumed to be running on resource-rich stationary computers that are embedded in the environment (e.g., Mocha PC[2]). In our preliminary work, all computing entities in a smart space are presumed to have priori knowledge about the presence of a context broker. In the future design, we will attempt to use one of the available service discovery infrastructure (e.g., Jini, UPnP) to improve system flexibility. Our centralized design of the context broker is motivated by the need to support small devices that have relatively limited resources available for context acquisition and reasoning. With the presence of a broker, small devices such as cellphones, PDA and watches can offload their burdens of managing context knowledge onto a resource rich context broker, including reasoning with context, detecting and resolving inconsistent context knowledge. Furthermore, in an open and dynamic environment, users may desire their personal contextual information to be kept in private. A centralized management of context knowledge makes easy to implement privacy protection and information security.

A centralized broker can be the "bottle-neck" in a distributed system, creating a single point of failure in the system. To address this problem, we plan to investigate and develop a fault-tolerance approach based on the Persistent Broker Team [14] approach. Our idea is to introduce a team of brokers in that each member has the responsibility to ensure at least one broker is available to provide services. In the case when the number of available team members falls below a predefined threshold (e.g., some broker becomes unreachable due to network failures), the remaining active team members will attempt to recruit or instantiate new brokers. In a broker team, the Joint Intention protocol [19] is used to bring about the mutual beliefs of the team states and team commitments.

A context broker has the following four functional components:

1. **Context Knowledge Base**: a persistent storage of the context knowledge. It provides a set of API's for other components in a broker to access the stored knowledge. It also contains the ontologies of a specific smart space (e.g., the ontologies of an intelligent meeting room) and some heuristic knowledge associated with the space (e.g., a company's daily operation hours are between 9:00 AM to 5:00 PM; no person can be physically present at two different meeting locations during the same time interval).

2. **Context Reasoning Engine**: a reactive inference engine that reasons over the stored context knowledge. Two types of inferences can take place in this engine: (i) inferences that use ontologies to deduce context knowledge, and (ii) inferences that use heuristic knowledge to detect and resolve inconsistent knowledge.

3. **Context Acquisition Module**: a library of procedures that forms a middle-ware abstraction for context acquisition. The role of this component is similar to the role of the Context Widgets in the Context Toolkit [8], which is to shield the low-level sensing implementations from the high-level applications.

4. **Policy Management Module**: a set of inference rules that deduce instructions for enforing user policies. Some rules are defined for deciding the right permissions for different computing entities to share a particular piece of context information, and some rules are defined for selecting the recipients to receive notifications of context changes.

5. PROTOTYPING AN INTELLIGENT MEETING ROOM

To demonstrate the feasibility of our Context Broker Architecture, we are using CoBrA to prototype an intelligent meeting room system called **EasyMeeting**, which provides assistants to meeting speakers, audiences, and organizers based on their situational needs. EasyMeeting is an extension to Vigil, a smart space system that we have previously developed [21]. Security is the main focus in Vigil. A role based access control mechanism is implemented in Vigil to allow users control to the permissions to access different services using policies. Vigil differs from other frameworks in using logic inference rules to reason about the rights of different users.

[2] `http://www.cappuccinopc.com/mochap4.asp`

Vigil has shown great promises in building flexible and secure smart spaces [21]. However, it lacks the necessary support for context-aware systems. To improve upon Vigil, in EasyMeeting we use OWL to represent context ontologies, and we exploit a context broker to support context reasoning.

In the rest of this section, first, we overview the ontologies that we have developed to support EasyMeeting and their potential role in the context reasoning, and second, we describe a prototype implementation of the context broker.

5.1 Ontologies

We have developed a set of ontologies for modeling context in an intelligent meeting room. This set of ontologies called COBRA-ONT defines typical concepts and relations for describing physical locations, time, people, software agents, mobile devices, meeting events.

5.1.1 Reasoning with the Physical Location Ontologies

Understanding the context associated with physical locations is extremely important in context-aware systems. An ontology of physical locations in COBRA-ONT includes the descriptions of places with identifiable geographic boundaries (e.g., rooms, buildings), places with spatial properties (e.g., atomic places, compound places), and places with temporal properties (e.g., meeting rooms during the working hours, offices on a public holiday). An ontology of the physical location context include geographic attributes, typical social norms of a particular place, objects that occupy or are contained in a particular space, and events that occur at a particular place.

In our ontology the class `Place` is the parent class of all represented place classes. Subclasses of `Place` are `Campus`, `Building`, `Room`, `Hallway`, `Parkinglot`, `Restroom`, and `ConferenceRoom`, which are concepts of places with identifiable geographic boundaries.

COBRA-ONT divids subclasses of `Place` into types of either atomic place or compound place, which are concepts of places with spatial properties. Atomic places (e.g., `ConferenceRoom`, `Hallway`) are places that cannot be defined to spatially subsume other places. Compound places (e.g., `Campus`, `Building`) are places that can spatially subsume other atomic or compound places[3].

Spatial containment inference is a type of reasoning with location context. Let us consider the scenario described in Section 4.1. As Alice enters the conference room, information acquired from the sensors in the room may lead the context broker to conclude that Alice is in the room R230. Because the broker has an ontology of the associated location (e.g., the Engineering building spatially subsumes the room R230, and the UMBC main campus spatially subsumes the Engineering building), the broker can draw new conclusions about Alice's location context, e.g., Alice is located in the Engineering building, Alice is located on the UMBC main campus.

[3]Note that present ontology do permit the definition of some illogical statements, e.g., a building spatially subsumes a campus (since they both are type of compound place). We are developing solutions to resolve this problem in the next version of the ontology

Reasoning with the spatial containment relation can also help the broker to detect errors in sensing. Let us assume in addition to the location ontology, the broker also has some heuristic knowledge about the associated location, for example, *"no person can be physically present at more than one atomic place during the same time interval"*. When coupled with the location ontology, this knowledge can help the broker to detect if there is any inconsistency about a user's location context. Imagine that due to sensing errors, some sensors falsely detect the present of Alice and informs the broker that she is located in the parking lot A. Since Alice is known to be located in the room R230, and both the parking lot A and the room R230 are atomic places, the broker can immediately conclude the location context of Alice is inconsistent with the known heuristic knowledge.

5.1.2 Reasoning with the Device Ontologies

In a pervasive computing environment, devices in the immediate vicinity of a user are also part of the user's context. Device context can include basic knowledge about the device profiles (e.g., does a particular device support color display?), the device ownership relation (e.g., who is the owner of a particular device?), temporal properties associated with a device (e.g., when was the last time a particular device has been used?), and spatial properties associated with a device (e.g., what is the distance between a particular device and the room in which its owner is currently in?).

To support reasoning with device profiles, COBRA-ONT includes an ontology of device hardware and software profiles. Parts of this ontology are adopted from the FIPA device ontology specification [9]. The hardware profile ontology includes concepts of screen displays (e.g., screen width and length, display color profile), device memory (e.g., amount of memory, memory size unit, and memory usage type), device network capability (e.g., support for wireless/wired communications, supported network interfaces). The software profile ontology includes concepts of device operating systems and supported computing platforms.

By extending the device profile ontology, COBRA-ONT provides an ontology of mobile devices. The goal is to define specific ontology classes that represent different types of mobile devices and properties that are associated with these devices. Represented types of mobile devices are SonyEricsson T68i, SonyEricsson T800, and Palm TungstenT. Defined properties include the hardware and software profiles of these devices and additional relations that associate the devices with people. For example, the `ownedBy` property expresses the relation between a device and its owner, and the `usedBy` property expresses the relation between a device and its user. Both of these properties have inverse properties (i.e., the `owns` and `uses` properties).

To illustrate how reasoning with device ontologies can play a role in context-aware systems, let us consider the following use case: after Alice enters the meeting room R230, her cellphone presents Alice's policy to the context broker in the room. Assuming the context broker has an ontology of the device that sends the policy, it reasons about the profile of the device, e.g., the sender is a type of SonyEricssonT68i cellphone, and its Bluetooth communication supports the OBEX

object push service for exchanging vCard contacts. Knowing the device profile, the context broker informs all meeting services that could take advantage of this information, e.g., a contact exchange service that can automatically push new contact information into the mobile devices that a meeting participant carries. Additionally, the context broker reasons about the person who owns and uses the device (e.g., the person Alice is the owner of this device, and no evidence shows the device is used by other people). Since Alice's SonyEricsson T68i cellphone is the room R230 and no evidence to the contrary, the context broker concludes Alice is also in the room R230.

5.1.3 Reasoning with the Temporal Ontologies

Aspects of context can also include temporal relations. To support reasoning with time and temporal relations, COBRA-ONT adopts the DAML-time ontology, which is a temporal ontology for expressing temporal aspects of the contents of web resource and for express time-related properties of web services [11]. The DAML-time ontology in COBRA-ONT is an OWL version of the original DAML-time ontology that is expressed in the DAML+OIL language[4].

The DAML-time ontology builds around a set of abstract temporal entities and temporal relation axioms. Our OWL representation of the ontology defines the vocabularies of the abstract temporal entities. However, it does not include a representation of the temporal relation axioms because the present OWL language does not have direct support for expressing axiomatic rules. Research in developing language constructs for representing rules in Semantic Web languages is underway [10].

In DAML-time the abstract temporal entity classes are `Instant` and `Interval`. Both are subclasses of the `TemporalEntity` class. A member of the `Instant` class represents an instant of time, which has associated temporal description properties that represent the concepts of second, minute, hour, day, month, year, and time zone. A member of the `Interval` class represents a time interval between two different time instants. Properties of this class include `beginOf` and `endOf`, which define the beginning time (a time instant) and the ending time (a time instant) of a time interval.

A type of relation between the individuals of the `TemporalEntity` class is temporal ordering. The temporal ordering relation can be expressed in the `before` and the `after` properties, i.e., an individual of the `Instant` or `Interval` class can have a `before` or `after` property value of another individual of `Instant` or `Interval` class. The temporal ordering relation can also be expressed using the `inside` and `time-between` properties, which describes a time instant is `inside` of a particular time interval, and a time interval is in `between` of two different time instants, respectively.

[4]To convert DAML-time from DAML+OIL to OWL, we have used the OWL Coverter, a tool for automating ontology conversion from DAML+OIL to OWL (`http://www.mindswap.org/2002/owl.html`)

The DAML-time ontology defines a number of predicates (properties) for linking time entities to events in the real world[5], which include `atTime`, expressing an event occurs at a particular time instant, `during`, express an event occurs during a particular time interval, and `holds`, expressing an event holds at a particular time instant or during a particular time interval.

To illustrate the use of temporal ontology, let us consider the following the use case: while Alice is in the room R230, different sensors notify the context broker of her presence. Each notification is linked to a particular time instant, e.g., an RFID sensor reports `atTime(locatedIn(Alice, R230), clockTime("13:04"))` and a facial recognition sensor reports `atTime(locatedIn(Alice, R230), clockTime("13:45"))`. As the predicate `clockTime` represents a time instant, the context broker can deduce the temporal ordering of the time during which Alice is located in the room R230, e.g., `during(locatedIn(Alice, R230), timeInterval("13:04-13:45"))`.

Knowing Alice is in the room R230 during a particular interval, the context broker can reason about her relation to the events in the room. For example, at `clockTime("13:55")`, the context broker learns that the room R230 is hosting a meeting which begins at `clockTime("13:00")`. From its knowledge about Alice's location (i.e., `during(locatedIn(Alice, R230), timeInterval("13:04-13:45"))`), the broker concludes that Alice is probably attending the meeting because `timeInterval("13:04-13:45")` is in between of the time instants `clockTime("13:00")` and `clockTime("13:55")`.

5.2 A Context Broker Prototype

We have implemented a context broker prototype to demonstrate its role in the EasyMeeting system. Our objective is to show the Semantic Web languages are adequate for dynamically constructing representations of context information acquired from sensors, and how this information then can be used to infer additional context knowledge. In our present implementation, a context broker can reason about the presence of people and device in an intelligent meeting room.

Figure 3 shows the design layout of our prototype system. Central to the system is a context broker. This broker is implemented as a FIPA compliant agent that runs on the JADE platform (a Java library for building FIPA compliant agents) [2]. The broker uses the Jena Semantic Web Toolkit[6] for managing and manipulating ontologies (e.g., dynamically constructing OWL ontology statements for agent communications, manipulating ontology knowledge that is stored in a persistent knowledge base).

RDQL (RDF Data Query Language) is used in the broker's reasoning engine to access the stored ontology knowledge. Using RDQL, the reasoning engine periodically queries the

[5]Descriptions of events are assumed to be defined by ontologies that are outside of the DAML-time ontology

[6]`http://www.hpl.hp.com/semweb/jena.htm`

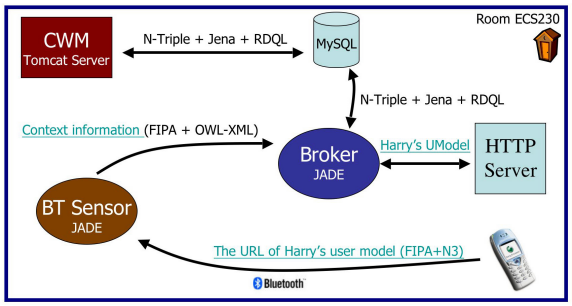

Figure 3: In our prototype system, the broker attempts to infer the location context of devices and users. An user model is dynamically acquired from an URL specified in the received user policy.

knowledge base for the presence of certain context knowledge (e.g., has any device been detected in the room, who is the owner of a particular device?). When queries return matched results, the broker automatically inserts new assertions about the local context into its knowledge base. For example, if queries return information about the presence of a new device and the person who owns the device, then the broker asserts the owner of the device is also present in the room.

For context sensing, the context broker delegates the tasks to other sensing agents in the environment. When the broker starts on a hosting JADE platform, it finds all sensing agents that are registered with the local yellow page service (FIPA Directory Facilitator) and sends a FIPA subscribe message to these agents, requesting to be notified about context changes. In our prototype system, we have implemented a sensing agent called BT Sensor, which is responsible for detecting Bluetooth OBEX object push events that are initiated by the mobile devices. As a mobile device sends an OBEX object to the BT Sensor (e.g., a SonyEricsson T68i cellphone sends a vNote object to the BT Sensor), the BT Sensor agent concludes the presence of the device and notifies the context broker.

Messages sent from a Bluetooth device to the BT Sensor agent contains a FIPA ACL message that informs the context broker of a user's background information (or user model). A user model includes a privacy policy that a user defines to control the use and the sharing of his/her context information. Due to the message size limitations in our Bluetooth devices, messages sent by the devices contain the URL of the web documents that have complete descriptions of the user models. The representation of this URL information is expressed in a RDF statement which is encoded in N3 [3], for example, `"agt:HarryChen agt:aboutMe http://umbc.edu/hchen4/aboutMe."`. The first term `agt:HarryChen` is a subject RDF resource that defines this statement is about the user Harry Chen. The second term `agt:aboutMe` is a property of the subject. The last term is the value of the property, which is an URL from which the user model of `agt:HarryChen` can be retrieved.

To show the underlying ontology reasoning in the broker, we have developed a web application, backed by the Apache Tomcat Server, for viewing the internal knowledge base of the context broker. In future, this web application will include administrator functions for manging a context broker (start, shutdown etc.).

6. RELATED WORK

In the past, a number of system architectures have been developed to support pervasive computing such as the Context Toolkit framework [17], Schilit's context-aware architecture [18], Cooltown [13], and Intelligent Room [7]. These systems have made progress in various aspects of pervasive computing but are weak in supporting knowledge sharing and context reasoning. A significant source of this weakness is their lack a common ontology with explicit semantic representation [6, 5].

Key differences between our architecture and the previous systems are the following:

- We use Semantic Web languages (i.e., RDF and OWL) to define ontologies of contexts, providing an explicit representation of contexts for reasoning and knowledge sharing. In the previous systems, contexts are often implemented as programming objects (e.g., Java class objects) or informally described in documentations.

- In CoBrA a resource-rich agent (i.e., the context broker) is provided to manage and maintain a shared model of context for all devices, services and agents in an associated space. In the previous systems, individual entities are required to manage and maintain their own context knowledge.

- The context reasoning in CoBrA gives context brokers the ability to infer new context knowledge (e.g., spatial relations, device profiles) that cannot be easily acquired from the physical sensors. In the previous systems, contexts acquired from sensors are presumed to be accurate and consistent.

- The use of policies in CoBrA allow users to control their contextual information, specifying the granularity of information that is shared by the systems and choosing recipients to receive notifications of their context changes. In preivous systems, acquired contextual information is allowed to be freely share by all computing entities in the environment, which could potentially jeopardize user privacy.

7. CONCLUSION & FUTURE WORK

The use of ontology is a key requirement for realizing pervasive context-aware systems. Our preliminary research in the Context Broker Architecture shows the Web Ontology Language OWL is adequate for defining ontologies for supporting context reasoning and knowledge sharing. As Semantic Web technologies and tools (i.e., programming libraries for manipulating ontologies and logic inference engines for ontology reasoning), we believe the Semantic Web will create new research opportunities for building pervasive context-aware systems.

At present the development of CoBrA and the EasyMeeting system is still in the early stage of research. Our short-term objective is to define an ontology for expressing privacy policy and to enhance a broker's reasoning with users and activities by including temporal and spatial relations. A part of our long-term objective is to deploy an intelligent meeting room in the newly constructed Information Technology and Engineering Building on the UMBC main campus.

REFERENCES

[1] R. Scott Cost andTim Finin andAnupam Joshi, Yun Peng, Charles Nicholas andHarry Chen, Lalana Kagal, Filip Perich andYouyong Zou andSovrin Tolia, and Ian Soboroff. Ittalks: A case study in the semantic web and daml. In *proceedings of the International Semantic Web Working Symposium*, July 2002.

[2] Fabio Bellifemine, Agostino Poggi, and Giovanni Rimassa. Developing multi agent systems with a fipa-compliant agent framework. *Software - Practice And Experience*, 31(2):103–128, 2001.

[3] Tim Berners-Lee. *Primer: Getting into RDF & Semantic Web using N3*, 2003.

[4] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific American*, may 2001.

[5] Harry Chen. An intelligent broker architecture for context-aware systems. PhD. dissertation proposal. See `http://users.ebiquity.org/~hchen4/phd/hcthsisp.pdf`, 2003.

[6] Harry Chen, Sovrin Tolia, Craig Sayers, Tim Finin, and Anupam Joshi. Creating context-aware software agents. In *Proceedings of the First GSFC/JPL Workshop on Radical Agent Concepts*, 2001.

[7] Michael H. Coen. Design principles for intelligent environments. In *AAAI/IAAI*, pages 547–554, 1998.

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

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

[10] Benjamin N. Grosof, Ian Horrocks, Raphael Volz, and Stefan Decker. Description logic programs: Combining logic programs with description logic. In *12th International Conference on the World Wide Web*, 2002.

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

[12] Lalana Kagal, Tim Finin, and Anupam Joshi. A policy language for a pervasive computing environment. In *IEEE 4th International Workshop on Policies for Distributed Systems and Networks*, 2003.

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

[14] Sanjeev Kumar, Philip R. Cohen, and Hector J. Levesque. The adaptive agent architecture: Achieving fault-tolerance using persistent broker teams. In *Proceedings of the Fourth International Conference on Multi-Agent Systems*, pages 159–166, 2000.

[15] Deborah L. McGuinness and Frank van Harmelen. Owl web ontology language overview. `http://www.w3.org/TR/owl-features/`, 2003.

[16] Nissanka B. Priyantha, Anit Chakraborty, and Hari Balakrishnan. The cricket location-support system. In *Mobile Computing and Networking*, pages 32–43, 2000.

[17] Daniel Salber, Anind K. Dey, and Gregory D. Abowd. The context toolkit: Aiding the development of context-enabled applications. In *CHI*, pages 434–441, 1999.

[18] Bill Schilit, Norman Adams, and Roy Want. Context-aware computing applications. In *IEEE Workshop on Mobile Computing Systems and Applications*, Santa Cruz, CA, US, 1994.

[19] Ira A. Smith and Philip R. Cohen. Toward a semantics for an agent communication language based on speech acts. In Howard Shrobe and Ted Senator, editors, *Proceedings of the Thirteenth National Conference on Artificial Intelligence and the Eighth Innovative Applications of Artificial Intelligence Conference, Vol. 2*, pages 24–31, Menlo Park, California, 1996. AAAI Press.

[20] Michael K. Smith, Chris Welty, and Deborah McGuinness. Owl web ontology language guide. `http://www.w3.org/TR/owl-guide/`, 2003.

[21] Jeffrey Undercoffer, Filip Perich, Andrej Cedilnik, Lalana Kagal, Anupam Joshi, and Tim 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.

[22] Frank van Harmelen, Jim Hendler, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein. Owl web ontology language reference. `http://www.w3.org/TR/owl-ref/)`, 2002.

[23] Roy Want, Andy Hopper, Veronica Falcao, and Jon Gibbons. The active badge location system. Technical Report 92.1, Olivetti Research Ltd., ORL, 24a Trumpington Street, Cambridge CB2 1QA, 1992.