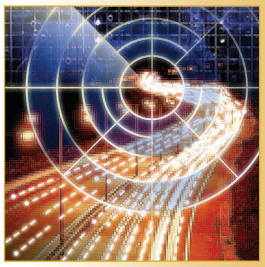
Editor: Michael N. Huhns • *huhns@sc.edu*



Intelligent Agents Meet the Semantic Web in Smart Spaces

A new smart meeting room system called EasyMeeting explores the use of multiagent systems, Semantic Web ontologies, reasoning, and declarative policies for security and privacy. Building on an earlier pervasive computing system, EasyMeeting provides relevant services and information to meeting participants based on their situational needs. The system also exploits the context-aware support provided by the Context Broker Architecture (Cobra). Cobra's intelligent broker agent maintains a shared context model for all computing entities in the space and enforces user-defined privacy policies.

Harry Chen, Tim Finin, Anupam Joshi, and Lalana Kagal
University of Maryland, Baltimore County

Filip Perich
Cougaar Software

Dipanjan Chakraborty
IBM India Research Laboratory

Pervasive computing (sometimes called ubiquitous computing) is a vision of our future computing lifestyle in which computer systems seamlessly integrate into our everyday lives, providing services and information "anywhere, anytime." Working toward this vision, researchers have developed a wide variety of pervasive computing applications and infrastructures. Pineau and colleagues,[1] for example, developed a mobile robotic assistant that provides reminding and guiding services to senior citizens with mild cognitive and physical impairments. Song and colleagues[2] created a ubiquitous computing infrastructure that lets users discover and compose services based on the tasks they intend to perform. Roman and colleagues[3] built a novel middleware infrastructure for managing resources and services in a user-centric active-space environment.

We have designed and implemented a pervasive computing system called EasyMeeting. It supports users in a smart meeting-room environment in which a distributed system of intelligent agents, services, devices, and sensors share a common goal: to provide relevant services and information to meeting participants on the basis of their contexts. By context, we mean a model of a location and its environmental attributes (such as temperature, noise level, and light intensity), and the people, physical objects, and computing entities it contains. This model necessarily extends to recognizing and representing the activities and tasks occurring in a location as well as the

 1089-7801/04/$20.00 © 2004 IEEE Published by the IEEE Computer Society

beliefs, desires, commitments, and intentions of the people and software agents involved.

A great challenge remains in defining an architecture that supports a community of context-aware agents in smart spaces. Critical research issues include modeling and reasoning (how to represent contextual information for machine processing and reasoning), knowledge sharing (how to enable agents to acquire consistent knowledge from unreliable sensors and agents), and user privacy protection (how to give users control of any private information the smart space acquires).[4] To address these issues, we built our EasyMeeting system using the Context Broker Architecture (Cobra), a middleware agent architecture for supporting context-aware agents in a smart space.[5]

Central to Cobra is an intelligent agent called the context broker. In a smart space, a context broker is responsible for:

- providing a centralized model of context that all devices, services, and agents in the space can share,
- acquiring contextual information from sources that are unreachable by the resource-limited devices,
- reasoning about contextual information that can't be directly acquired from the sensors,
- detecting and resolving inconsistent knowledge stored in the shared context model, and
- protecting privacy by enforcing policies that users have defined to control the sharing and use of their contextual information.

Cobra differs from other similar architectures because it uses ontologies expressed in the Web Ontology Language (OWL) to support context modeling and knowledge sharing, detect and resolve inconsistent context knowledge, and protect the user's privacy.

The EasyMeeting System

EasyMeeting is an extension of Vigil,[6] a third-generation pervasive-computing infrastructure developed at the University of Maryland, Baltimore County (UMBC). Although the research development behind Vigil shows great promise in building flexible and secure smart spaces,[7] it lacks the necessary support for context awareness and privacy protection.[8] To improve on the previous system, we added context-aware support to EasyMeeting by exploiting Cobra. Figure 1 illustrates the EasyMeeting system's architecture.

Vigil

The Vigil computing environment comprises clients, services, and Vigil managers, which are specialized server entities that facilitate system communication, client-role management, and service-access control. Computer applications provide services to clients — either human users or computing services — that use or receive such services in a smart space.

Services usually register themselves with one or more service managers, which provide directory listings of what's on offer and notify clients about status changes. Vigil uses a role-based inferencing mechanism to control access to services. When a service registers with a manager, the service specifies a list of roles that have permission to access it. This list of information is called *role-permission definition*. To prove the information's authenticity, the service manager requires the role-permission definition to be enclosed in a signed digital certificate. Upon receiving and verifying this certificate, the service manager adds the enclosed role-permission to its knowledge base.

To access a registered service, a client must first request a role-permission handle — a signed digital certificate — from the service manager and prove that it fulfills one of the service-defined roles. To do so, the client usually presents a role certificate generated by a role-assignment manager, which is a trusted server entity that can reason about a particular client's role from predefined system policy rules. The reasoning of the role-assignment manager is built on the Rei framework, which is a role-based policy reasoning engine.[9] A key feature of Rei is its use of *deontic concepts* (rights, prohibitions, obligations, and dispensations) to construct logic-inference rules. These are necessary because the system allows clients to delegate their right to access a particular service to other clients; a service manager can also revoke a client's right to access restricted services, and services can prohibit certain clients from accessing their services.

EasyMeeting Services

EasyMeeting's current implementation provides context-aware services for assisting speakers and audiences during presentations. We've already developed six services:

- *Speech understanding* recognizes predefined voice input vocabularies (such as "yes," "no," or

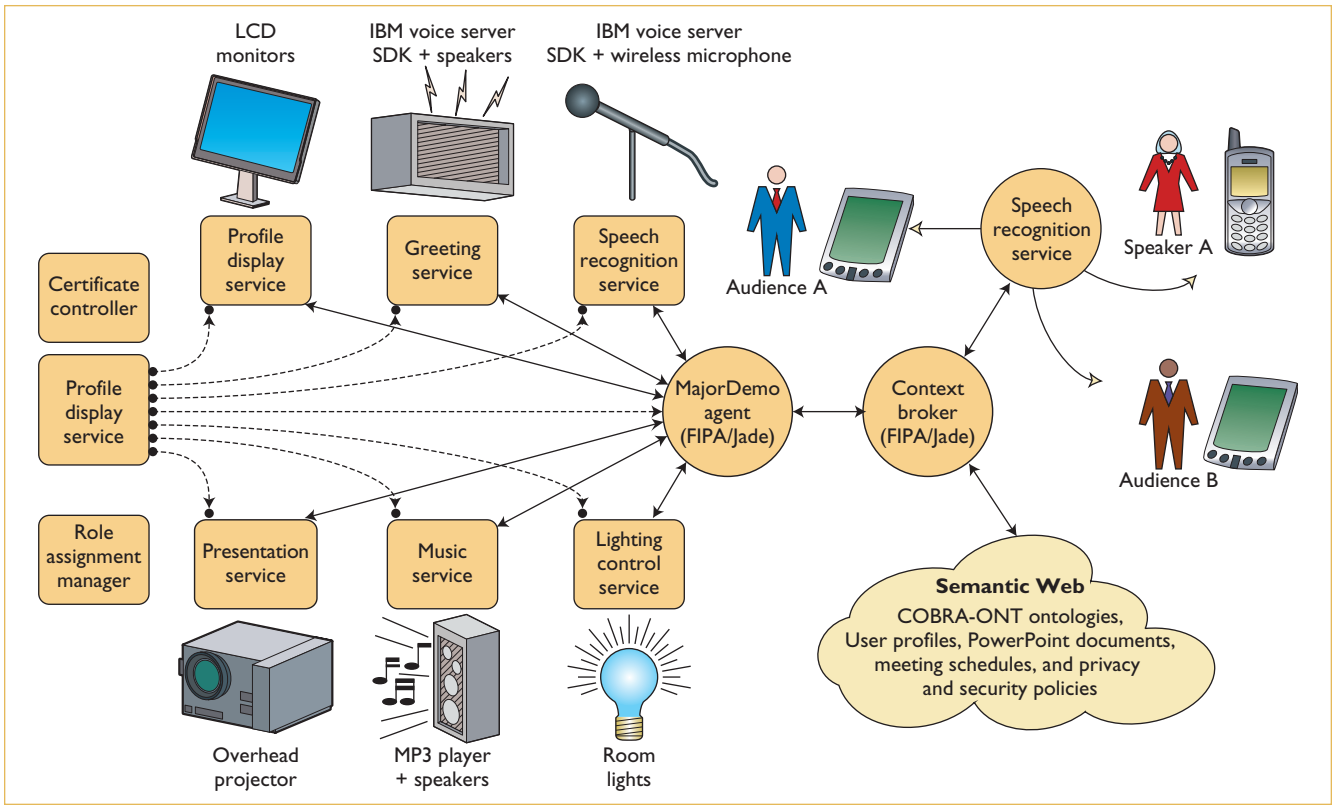


Figure 1. EasyMeeting architecture. The context broker shares its contextual knowledge with the MajorDemo agent. Using this knowledge, MajorDemo selects and then invokes appropriate services from the Vigil pervasive-computing infrastructure to provide relevant services and information to speakers and audiences.

"show Harry's presentation") and generates Centaurus Capability Markup Language (CCML)[10] commands for controlling other Vigil services.

- *Presentation* displays PowerPoint presentations on an overhead projector in the room and controls the flow of the presentation slides. It can fetch presentation files from a client-specified URL, and it invokes the corresponding PowerPoint script commands from CCML commands ("next," "back," "stop," "start").
- *Lighting control* adjusts a meeting room's lighting conditions.
- *Music* plays Web-accessible audio files, typically a continuous background-music loop prior to the meeting's official start.
- *Greeting* plays a specified audio file as a greeting message (such as "Welcome to the eBiquity Group, president Hrabowski").
- *Profile display* instructs all subscribed Web browsers to display URLs with speakers' background information on the audience's individual handheld devices.

Our implementation of the speech understanding and greeting services uses the IBM WebSphere Voice Server SDK and Voice XML. The presentation service's flow-control mechanism is implemented via PowerPoint's existing AppleScript commands. The lighting control service uses X10 technology. The music service is implemented with existing MP3 music player software. Finally, the profile display service is implemented as a Web-based server application that communicates with system components via a set of predefined network sockets.

Context-Awareness

Exploiting the notion of meeting context is a key feature in EasyMeeting. This information helps computer systems decide what services to provide based on the meeting participants' needs.

A central element in Cobra is the presence of a context broker, an intelligent agent that runs on a resource-rich stationary computer in the space. In EasyMeeting, the context broker builds and updates a shared context model and makes it available to appropriate agents and services. In particular, it acquires and maintains consistent knowledge about the meeting participants' indi-

vidual locations, scheduled events and presentations, speaker profiles, and the state of the meeting. The broker discovers and downloads some contextual information expressed in OWL directly from the Web or from data provided by sensing agents. It uses logical reasoning to infer contextual information that it can't directly acquire.

To facilitate the interactions between the Vigil system's services and Cobra's context broker, we introduced the MajorDemo agent (see Figure 1). This agent's role is to decide when and what services should be provided to meeting participants. It relies on the broker to provide information about the meeting context and uses the registered Vigil services to facilitate different meeting-related tasks.

Let's look at a typical EasyMeeting use case. On 8 September 2004, a presentation is scheduled to take place from 1:00 to 2:30 p.m. in Room 338, which is a smart meeting room. Moments before the event starts, the room's context broker acquires the meeting's schedule, which is expressed in OWL, from the Web and concludes that the meeting is about to start. As the meeting participants begin to arrive, the room's Bluetooth sensing agent detects the presence of different Bluetooth-enabled devices (such as cell phones and PDAs). Because each device has a unique device profile represented in a standard device ontology, the sensing agent can share this information with the broker.

Based on the user profile information stored in the broker's knowledge base (without having any evidence to the contrary), the context broker concludes that the owners of the detected devices are also located in Room 338. Among the present participants, Harry (the speaker) and president Hrabowski (the distinguished audience) are two people listed in the meeting schedule. The broker shares their location information with the subscribed MajorDemo agent.

Knowing that president Hrabowski has a distinguished-audience role, the MajorDemo agent invokes the greeting service. At 1:00 p.m., the context broker informs the MajorDemo agent that all listed key participants have arrived and that the presentation can start. Knowing that all the lights in the meeting are currently switched on and the background music is still playing, the agent invokes the `dim light` method on 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 via a wireless microphone — "load Harry's presentation." The voice-recognition service receives his voice input, and it generates a corresponding CCML command. The MajorDemo agent sends this command to the presentation service along with the URL at which the audience can download Harry's presentation (the context broker provides this information). As the presentation service loads Harry's PowerPoint slides, the MajorDemo agent invokes the profile-display service to show Harry's home page. Moments later, all the LCD displays sitting on the conference table start showing Harry's biosketch and profile. Using the same wireless microphone, Harry speaks to the system to control his presentation.

Cobra

Figure 2 shows a system diagram of the Cobra design. All computing entities in a smart space are presumed to have prior knowledge about the broker's presence, and all agents are presumed to communicate with the broker via the standard FIPA Agent Communication Language and the ontologies defined by Cobra.

The context broker consists of several components:

- *Context knowledge base*. All context knowledge is represented in Resource Description Framework (RDF) triples (subject, predicate, and object), which are stored in a persistent relational database backed by the Jena 2 Semantic Web framework.
- *Context-reasoning engine*. A rule-based inference engine defines how to deduce context knowledge via the ontology inference based on OWL's semantics and from domain-heuristic rules.
- *Context-acquisition module*. The goal of this library of procedures for acquiring contextual information is to create a middleware abstraction to hide low-level complexity in context acquisition.[11]
- *Policy-management module*. Before the context broker shares a user's information with another agent, it uses this module — its "conscience" — to determine the other agent's right to receive this knowledge; the broker shares the information only if the user-defined policy allows it.

The context-reasoning engine uses Jena's reasoning API to support inferences licensed by the OWL ontology and the Jess rule-based engine for other domain-specific reasoning. The present context-

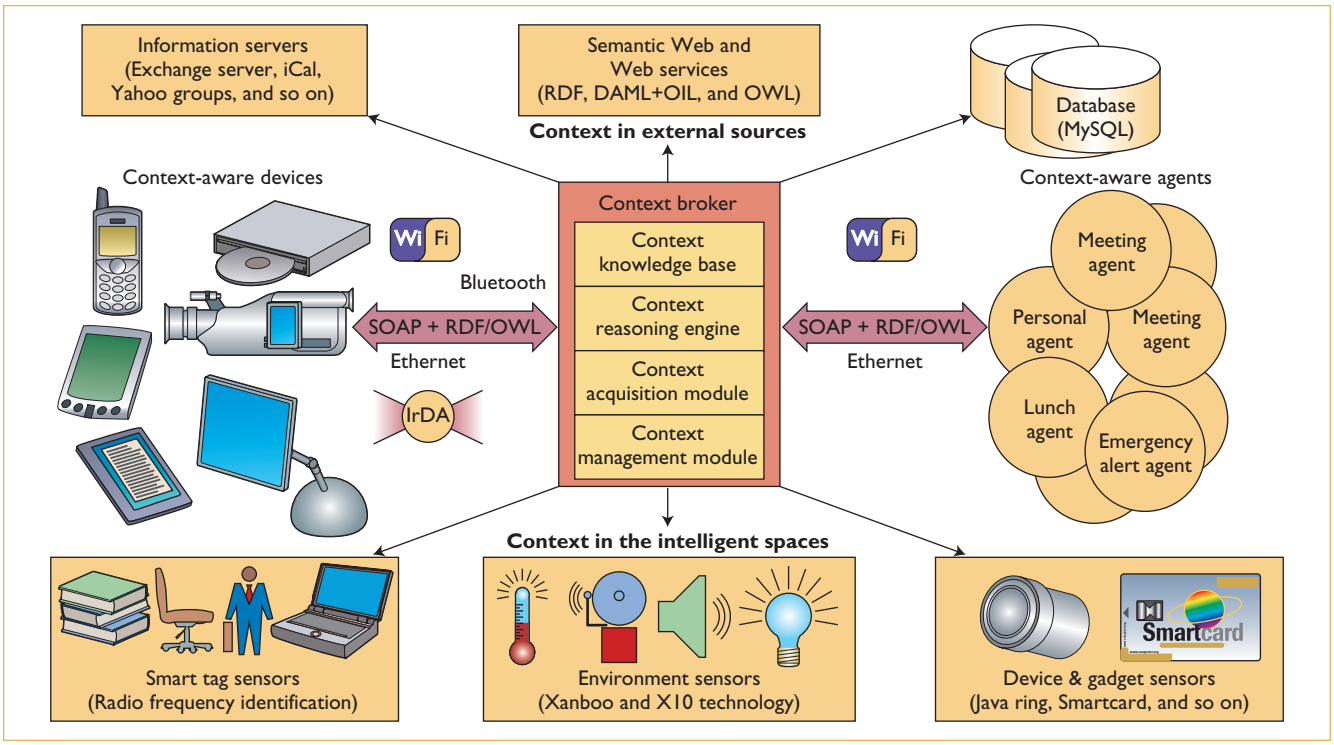


Figure 2. The Context Broker Architecture (Cobra). A context broker acquires contextual information from heterogeneous sources and fuses it into a coherent model to share with computing entities sharing the same space.

acquisition module consists of implementations for detecting the presence of Bluetooth devices by sensing their Bluetooth MAC addresses.

Semantic Web Ontologies in Cobra

Cobra differs from other frameworks in its use of Semantic Web languages both to express context ontologies and to reason about contexts.[12] We chose OWL for Cobra for several reasons. Because OWL is a reasonably expressive knowledge-representation language, it's suitable for defining many of the general ontologies used in building intelligent pervasive computing applications. Moreover, ontologies expressed in OWL have a normative syntax in RDF and XML that can be mapped to several other forms of surface syntax, which offers flexibility in processing, displaying, and storing ontology data. Finally, OWL is expressly designed as an ontology language, and it has many predefined classes and properties useful for expressing ontological information.

The Cobra ontology (Cobra-Ont) imports several different upper ontologies from the Standard Ontology for Ubiquitous and Pervasive Applications (SOUPA). The Semantic Web in UbiComp Special Interest Group (http://pervasive.semanticweb.org) developed SOUPA to support ontology-driven pervasive computing applications. From the SOUPA ontology, Cobra-Ont imports vocabularies for expressing time, space, policy, social networks, actions, location context, documents, and events. Figure 3 (next page) illustrates the SOUPA ontology's structural organization.

Let's look at some examples that use COBRA-ONT to express and reason about user profiles, meeting events, and temporal and spatial relations.

User profiles. A typical user profile consists of information that describes the person's

- background (contact information, employment information, professional associations, and preferences),
- social relations (friends and coworkers),
- mobile-device or personal-agent profiles (the type of communication interfaces and display resolutions a device supports or a personal agent's ID), and
- daily meeting schedule and associated roles and preferences (meeting times and locations, slides for the person's typical presentations, and so on).

In EasyMeeting, a person's user profile is publicly

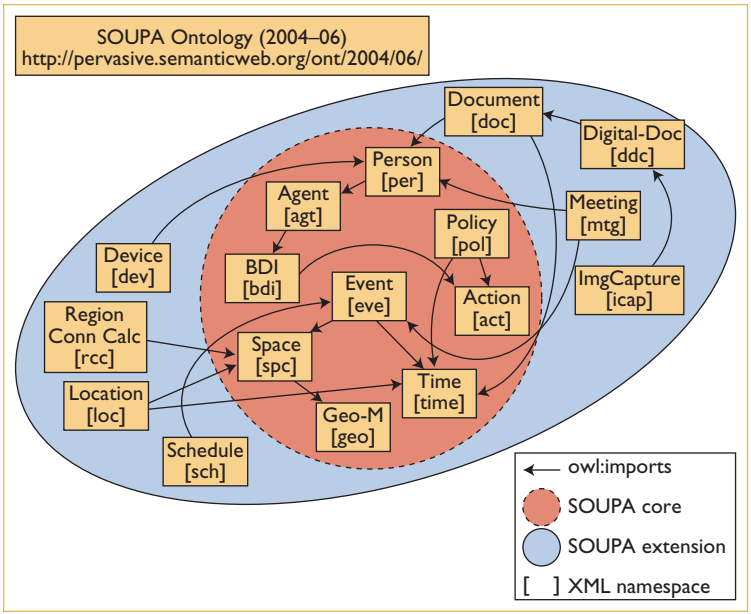


Figure 3. SOUPA consists of two sets of ontology documents: SOUPA Core and SOUPA Extension. The OWL construct enables a modular design of the ontology. Different domain vocabularies are grouped under different XML namespaces.

accessible through his or her homepage. Figure 4 (next page) shows a partial description of Harry Chen's user profile, the beginning of which defines typical social-network information (including his contact information and some of the people he knows). The profile also defines the `agt:intends` property, which describes a meeting Harry plans to attend. Note that the associated meeting's details aren't specified here; the agents that will process the profile presumably have other means of acquiring this information. When this information becomes available to the agents, the meeting's RDF resource description URI can help them relate the person described in the profile to various meeting properties and infer whether Harry Chen is a speaker or an attendee.

The last part of Figure 4 describes the profile for Harry's cell phone, including the model, serial number, and Bluetooth MAC address. This information can help the context broker reason about the location of the phone's owner when it detects the phone's location.

Meeting schedules. The context broker can reason about a meeting's various properties from its scheduled events, including the temporal state of the event, the participants, and their respective roles. A schedule's ontological document typically contains information about the type of meeting, its location, starting and ending times, and a list of expected participants.

In EasyMeeting, meeting schedule information is publicly accessible via a predefined meeting event announcement Web site. When describing an event schedule, the event's location and the properties are expressed with vocabularies from the SOUPA space and time ontologies.

Time and space. COBRA-ONT imports the SOUPA time and space ontologies, which are extremely useful for reasoning about context.[12] Time ontologies, for example, can help a context broker reason about the temporal orders among different events in a meeting room, whereas space ontologies can help the broker reason about a person's location context.

The SOUPA space ontology supports reasoning about the spatial relations between various types of geographical regions, mapping from geospatial coordinates to the symbolic representation of space and vice versa, and geographical measurements of space.

In the symbolic representation model, the `spc:SpatialThing` class represents a set of all things that have spatial extensions in the SOUPA domain. All the spatial things typically found in maps or construction blueprints are called `spc:GeographicalSpace`. This class is defined as the union of the `spc:GeographicalRegion`, `spc:FixedStructure`, and `spc:SpaceInAFixedStructure` classes.

An individual member of the `spc:GeographicalRegion` class typically represents a geographical region controlled by some political body (for example, the US is controlled by the US government). This relation is expressed by the `spc:controls` property, the domain of which is `spc:GeopoliticalEntity` and the range of which is `spc:GeographicalRegion`. Knowing which political entity controls a particular geographical region, a pervasive computing system can choose to apply the appropriate policies defined by the political entity to guide its behavior — a system could apply different sets of privacy protection schemes based on policies defined by the local political entities.

To support spatial containment reasoning, individual members of the `spc:GeographicalSpace` class can relate to each other through the `spc:spatiallySubsumes` and `spc:spatiallySubsumedBy` properties: a country region could spatially subsume a state region, a state

```
<foaf:Person>
  <foaf:name>Harry Chen</foaf:name>
  <foaf:title>Mr</foaf:title>
  <foaf:firstName>Harry</foaf:firstName>
  <foaf:surname>Chen</foaf:surname>
  <foaf:nick>hchen1</foaf:nick>

  <per:gender rdf:resource="&per;Male"/>


<foaf:mbox_sha1sum>944880c478f58ac2a5a63
fa3e922e712a0e327fc</foaf:mbox_sha1sum>

  <foaf:homepage
rdf:resource="http://umbc.edu/people/
hchen4"/>

  <foaf:depiction
rdf:resource="http://ebiquity.umbc.edu/
v2.1/faces/136.jpeg"/>

  <foaf:workplaceHomepage
rdf:resource="http://ebiquity.umbc.edu"
/>

  <foaf:schoolHomepage
rdf:resource="http://www.umbc.edu"/>

  <foaf:knows>
    <foaf:Person>
      <foaf:name>Tim Finin</foaf:name>

<foaf:mbox_sha1sum>9da08e2b4dc670d9254ab
4a4b4d61637fed3b18f</foaf:mbox_sha1sum>
      <rdfs:seeAlso
rdf:resource="http://www.cs.umbc.edu/
~finin/finin.rdf"/>
    </foaf:Person>
  </foaf:knows>

  <agt:intends>
    <pmt:ParticipateMeeting>
      <pmt:meeting
rdf:resource="http://ebiquity.umbc.edu/m
eeting/2004#aDemoSession"/>
    </pmt:ParticipateMeeting>
  </agt:intends>

  <aca:oftenUsedSlides
rdf:resource="http://umbc.edu/~hchen4/pp
t/f1"/>
</foaf:Person>

<dev:NokiaCellphone>
  <dev:hasUser>
    <foaf:Person>

<foaf:mbox_sha1sum>944880c478f58ac2a5a63
fa3e922e712a0e327fc</foaf:mbox_sha1sum>
      <foaf:homepage
rdf:resource="http://umbc.edu/people/
hchen4"/>
    </foaf:Person>
  </dev:hasUser>

  <dev:modelNumber
rdf:datatype="&xsd;string">3650</dev:mod
elNumber>
  <dev:serialNumber
rdf:datatype="&xsd;string">351102/50/380
59/8</dev:serialNumber>
  <dev:bluetoothMAC
rdf:datatype="&xsd;string">00-60-57-5a-
a0-21</dev:bluetoothMAC>
</dev:NokiaCellphone>
```

Figure 4. User profile. The vocabularies used to describe Harry Chen's social network and personal device are imported from the SOUPA and FOAF (friend-of-a-friend) ontologies.

region could spatially subsume a building, and a building could spatially subsume a room. Thus, knowing the room in which a device is located, we can infer the building, the state, and the country that spatially subsumes that room.

Context Reasoning in Cobra

In its current implementation, the context broker's ontology reasoning is backed by the Jena rule engine and its Java API (http://jena.sourceforge.net). To reason about contextual information that can't be inferred from ontology axioms alone, the context broker uses a forward-chaining inference procedure defined in the Java Expert System Shell (Jess).

Context-reasoning algorithm. Rules defined in Jess are executed as part of the context broker's reasoning implementation. 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 (such as a person's location or a meeting's state). If such information is unknown, the broker decides whether it can infer this type of context using only ontology reasoning. If logic inference is required, the context broker attempts to find all essential supporting facts by querying the ontology model. The broker then converts the RDF representation of the facts into the corresponding Jess representation and asserts them into the Jess engine. After executing the predefined forward-chaining procedure, the broker adds the corresponding RDF representations into the ontology model for any new facts it can deduce.

Assumption-based reasoning. The current implementation of our logical-inference procedure is rigid, and we're investigating an assumption-based reasoning approach[13] to improve flexibility. A prototype implements a reasoner based on David Poole's Theorist framework,[14] which is a Prolog meta-interpreter for processing assumption-based reasoning. Unlike conventional deductive-reasoning systems, this framework's logic inference consists of both facts (axioms given as true) and assumptions (instances of the possible hypotheses that can be assumed if they're consistent with the facts).

One way to use Theorist in Cobra is for *context* reasoning, which exploits both default reasoning, a type of nonmonotonic reasoning that allows a conclusion to be reached in the absence of any reason to doubt it, and *abductive* reasoning, the process of using inference to find the best explanation. When the broker receives an *observation* (all contextual information the broker acquires about the environment), it first uses abduction to determine possible causes and then uses default reasoning to predict what else will follow from them.[13]

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 device's owner is also in that room. Hypotheses `H2` states that if the system knows someone is in a room where a meeting is scheduled to take place, and that person is a speaker at the meeting, and if no evidence shows that the person is not in that room, then the system should infer the person intends to give a presentation at the meeting. Fact `F1` states that cell-phone T68i is located in Room RM338; facts `F2`, `F3`, and `F4` state that Harry is the owner of T68i, that meeting m1203 is scheduled to take place in Room RM338, and that Harry is a speaker at meeting m1203, respectively. We expect `F1` to be knowledge acquired from the sensors and `F2`, `F3`, and `F4` to be knowledge acquired from the OWL descriptions of the person profiles, device profiles, and meeting schedules published on the Web.

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

Privacy Protection in Cobra

In EasyMeeting, several privacy issues must be addressed:

- Users might be unaware that their trusted context broker can share private information about them with services they don't trust.
- It is infeasible to require users to manually define privacy-protection rules for all the possible contextual information the context broker can collect.
- Although users want to hide most of their private information from untrusted agents, they often can't completely prohibit information from being shared if they want to receive context-aware services.

Thus, we need a way to let users adjust the granularity of the information to be shared.

Figure 5 illustrates the SOUPA policy ontology's structural organization. Using the SOUPA policy ontology, users can define customized policy rules to permit or forbid access to their private information. To compute the permissions defined by a user

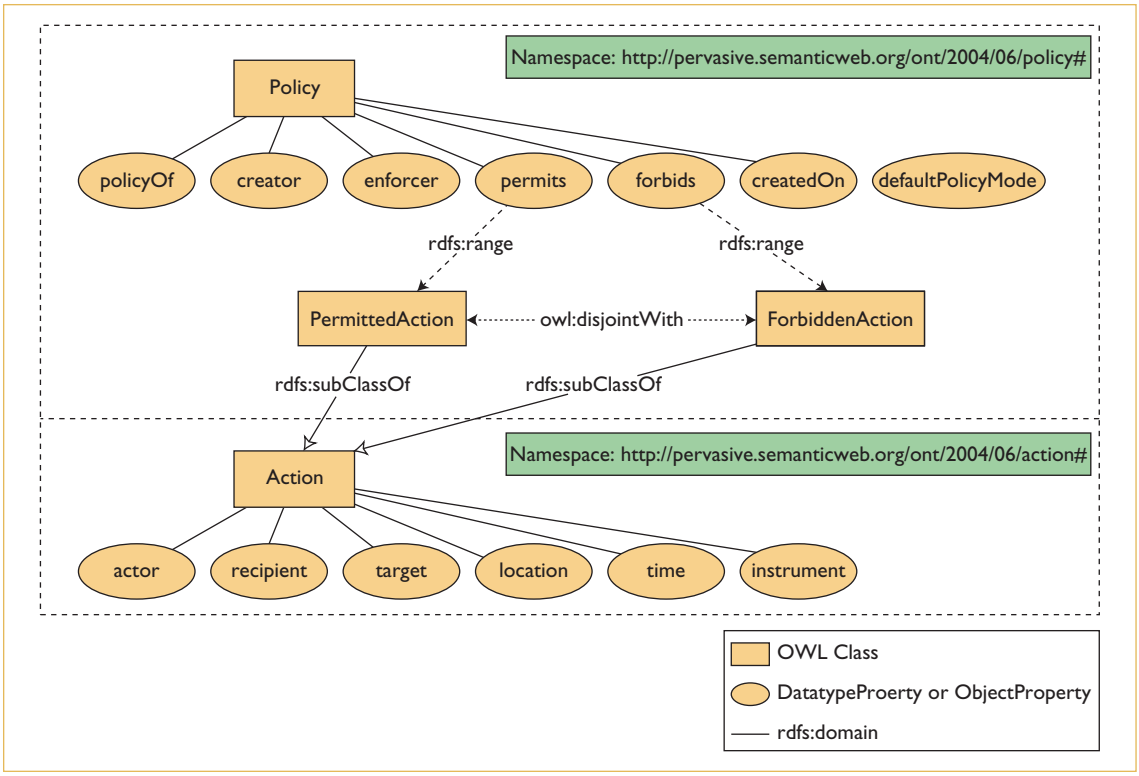


Figure 5. The SOUPA policy ontology. By importing SOUPA's action and time ontologies, we can define the vocabularies for describing rules that permit or forbid different agents' access.

policy, the context broker uses a policy-reasoning algorithm that exploits the description logic inference over OWL. To complement the user-defined policy with a global policy and let users adjust their information's granularity, we also implemented a metapolicy-reasoning mechanism suggested by the SOUPA ontology in the context broker. Figure 6 (next page) shows an example of a policy expressed in RDF's Notation3 syntax (www.w3.org/DesignIssues/Notation3.html).

Feedback from Demonstrations

As part of the rapid prototyping methodology we adopted, we demonstrated the EasyMeeting system to three groups of people. Our goal was to get feedback from users outside of our research team and observe their experiences in a context-aware smart meeting-room environment. The groups of users we invited to our demonstrations included UMBC university administrators and visitors from commercial companies and other universities.

We began each session with a brief introduction to the system's underlying infrastructure and the tasks the users were expected to perform. During the initial stage, we set up distinctive user profiles for each participant and posted these ontological documents on the Web. We also gave each participant a Bluetooth-enabled mobile device and asked them to activate the Bluetooth connection before arriving at the meeting room. In addition, we published a document with OWL annotations that described the meeting schedule on the Web. After all the participants arrived, the smart meeting room dimmed the lights and turned off the background music. The speakers were encouraged to ask the system to load presentations (described in their profiles).

In general, all the testers' feedback was positive. Most were excited about the concept, but they also raised several critical issues that pointed to key weaknesses in our current system. First, the system has a limited ability to handle unexpected situational changes. The absence of anticipated participants, for example, affected the context broker's logic reasoning. Second, the workflow process was too rigid and could be unsuitable for everyday usage — the current system doesn't take into consideration that a meeting can be paused and resumed, for example, and thus can't implement specific behavior to handle such situations. Finally, some users argued that merely using policy to control how private information is shared

```
# For a complete and more elaborate
policy example, see also
#
http://cobra.umbc.edu/ont/2004/05/
harrychen-policy

<http://umbc.edu/~hchen4/hchen.pol> a
pol:Policy;
  pol:policyOf  [ a per:Person; per:name
"Harry Chen"^^xsd:string ]
  pol:defaultPolicyMode pol:Conservative;

  # Rule 1: all individuals of CLS2 are
permitted actions#
  pol:permits ha:CLS2;

  # Rule 2: all individuals of CLS3 are
forbidden actions#
  pol:forbids ha:CLS3.

ha:CLS2 a :Class;

   rdfs:comment "Share my location
information with the ebiquity group
members iff
     the location information describes
me being in ITE210A, ITE325B or on the
UMBC campus.";

   :intersectionOf  (
      ebact:ShareLocationInfo
      [ :allValuesFrom
ebm:EbiquityMember; :onProperty
act:recipient ]
      [ :onProperty act:target;
:someValuesFrom
ha:MyRestrictedLocationContext ] ) .

ha:CLS3 a :Class;
   rdfs:comment "Share my location
information with untrusted service
agent";
   :intersectionOf  (
      ebact:ShareLocationInfo
      [ :allValuesFrom
ha:UntrustedServiceAgent; :onProperty
act:recipient ] ) .

ha:MyRestrictedLocationContext a :Class;
  :intersectionOf  (
     loc:LocationContext
     [ :onProperty loc:boundedWithin;
:someValuesFrom ha:foo-a1 ] ) .

ha:foo-a1 a :Class;
  :oneOf  (ebgeo:ITE210A  ebgeo:ITE325B
ebgeo:UMBCMainCampus).

ha:UntrustedServiceAgent a :Class;
rdfs:subClassOf agt:Agent;
    :oneOf  (
       <http://www.orbitz.com#locTrack>
<http://www.foobar.com#whereRu>
       <http://www.foofoobar.com#abc>  )
.
```

Figure 6. The SOUPA policy expressed in N3 notation. This policy allows some of Harry Chen's location information to be shared with the members of the eBiquity Group, but it also forbids any location information from being shared with untrusted agents. The action classes `CLS2` *and* `CLS3` *describe the types of actions the individual policy rule regulates.*

doesn't address other kinds of privacy concerns such as the logging and persistent storage of a user's private information by the agents, and the possibility for the agents acquiring certain private user information by reasoning over an aggregated collection of their public information. We plan to address these issues in the future.

Conclusions

The EasyMeeting and Cobra prototypes we've developed demonstrate the feasibility of using OWL ontologies to let distributed agents share knowledge, reason about contextual information, and express policies for user privacy protection. Although the initial results of our informal user evaluation shows that users are enthusiastic about context-aware services in a smart meeting room environment, we have many challenging research issues yet to address. These challenges include the scalability of knowledge sharing in a distributed and dynamic environment, the performance and time complexity of context reasoning in the presence of a vast amount of sensing data, and the user-interface issues associated with editing and maintaining user privacy policies. One of our short-term objectives is to optimize our existing implementation for a long-term use-case study within the UMBC Computer Science and Electrical

Engineering Department, using the EasyMeeting system as a common platform for building new services and intelligent agents. In the future, we plan to expand our context-aware support to include tracking an absent participant's location, tracking the availability of a portable projector device, and exchanging contact information between attendees.

References

1. J. Pineau et al., "Towards Robotic Assistants in Nursing Homes: Challenges and Results," *Robotics and Autonomous Systems*, vol. 42, nos. 3 and 4, 2003, pp. 271–281.
2. Z. Song, Y. Labrou, and R. Masuoka, "Dynamic Service Discovery and Management in Task Computing," *Proc. 1st Ann. Int'l Conf. Mobile and Ubiquitous Systems*, IEEE Press, 2004, pp. 310–318.
3. M. Roman and R.H. Campbell, "Gaia: Enabling Active Spaces," *Proc. 9th ACM Special Interest Group on Operating Systems (SIGOPS) European Workshop*, ACM Press, 2000, pp. 229–234.
4. M. Tambe, P. Scerri, and D.V. Pynadath, "Adjustable Autonomy for the Real World," *Proc. AAAI Spring Symp. Safe Learning Agents*, AAAI Press, 2002, pp. 43–53.
5. H. Chen, T. Finin, and A. Joshi, "Semantic Web in the Context Broker Architecture," *Proc. Int'l Conf. Pervasive Computing and Comm.* (PerCom 04), IEEE Press, 2004, pp. 277–286.
6. A.K. Dey, *Providing Architectural Support for Building Context-Aware Applications*, PhD thesis, College of Computing, Georgia Inst. of Technology, 2000.
7. T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web," *Scientific Am.*, May 2001, pp. 34–43.
8. J. Undercoffer et al., "A Secure Infrastructure for Service Discovery and Management in Pervasive Computing," *J. Special Issues on Mobility of Systems, Users, Data, and Computing*, vol. 8, no. 2, 2003, pp. 113–125
9. L. Kagal, T. Finin, and A. Joshi, "A Policy Based Approach to Security for the Semantic Web," *Proc. 2nd Int'l Semantic Web Conf.* (ISWC 03), Springer-Verlag, 2003, pp. 402–418.
10. L. Kagal et al., "Centaurus: A Framework for Intelligent Services in a Mobile Environment," *Proc. Int'l Workshop on Smart Appliances and Wearable Computing*, IEEE Press, 2001, pp. 195–201.
11. J. Carroll et al., *Jena: Implementing the Semantic Web Recommendations*, tech. report HPL-2003-146, Hewlett-Packard Labs, 2003.
12. H. Chen, T. Finin, and A. Joshi, "An Ontology for Context-Aware Pervasive Computing Environments," *Knowledge Eng. Rev.*, vol. 18, no. 3, 2004, pp. 197–207.
13. A.K. MacKworth, R.G. Goebel, and D.I. Poole, *Computational Intelligence: A Logical Approach*, Oxford Univ. Press, 1998, pp. 319–342.
14. D. Poole, "Compiling a Default Reasoning System into Prolog," *New Generation Computing*, vol. 9, no. 1, 1991, pp. 3–38.

Harry Chen is a PhD candidate in the Department of Computer Science and Electrical Engineering at the University of Maryland, Baltimore County (UMBC). His technical interests include pervasive computing, the Semantic Web, multiagent systems, privacy, and security. Chen received an MS in computer science from the University of Maryland, Baltimore County. He is a member of the AAAI, the ACM, the IEEE, Upsilon Pi Epsilon, and Golden Key National Honor Society. Contact him at harry.chen@umbc.edu.

Tim Finin is a professor in the Department of Computer Science and Electrical Engineering at UMBC. His technical interests include multi-agent systems, the Semantic Web, and mobile and pervasive computing. Finin received a PhD in computer science from the University of Illinois at Urbana-Champaign. Contact him at finin@umbc.edu.

Anupam Joshi is an associate professor in the Department of Computer Science and Electrical Engineering at UMBC. His technical interests include mobile and networked computing, intelligent systems, data mining, and the Semantic Web. Joshi received a PhD in computer science from Purdue University. He is a member of the ACM, the IEEE, and the IEEE Computer Society. Contact him at joshi@cs.umbc.edu.

Filip Perich is a senior research engineer at Cougaar Software. His primary research focus is on data management in distributed systems, particularly in mobile ad hoc networks. Perich received a PhD and an MS in computer science from UMBC and a BA in mathematics from Washington College. He is a member of the ACM and the MAA. Contact him at filip.perich@umbc.edu.

Dipanjan Chakraborty is a research staff member at IBM India Research Laboratory. His technical interests include mobile and pervasive computing environments, mobile and e-commerce, and peer-to-peer systems. Chakraborty received an MS and a PhD in computer science from the University of Maryland, Baltimore County. He is a member of the IEEE. Contact him at dchakr1@cs.umbc.edu.

Lalana Kagal is a PhD candidate in the Department of Computer Science and Electrical Engineering at UMBC. Her technical interests include artificial intelligence, security, and mobile computing. Kagal received a BS and an MS in computer science from Pune University, India. Contact her at lkagal1@cs.umbc.edu.