SOUPA: Standard Ontology for Ubiquitous and Pervasive Applications*

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

Abstract

We describe a shared ontology called SOUPA – Standard Ontology for Ubiquitous and Pervasive Applications. SOUPA is designed to model and support pervasive computing applications. This ontology is expressed using the Web Ontology Language OWL and includes modular component vocabularies to represent intelligent agents with associated beliefs, desires, and intentions, time, space, events, user profiles, actions, and policies for security and privacy. We discuss how SOUPA can be extended and used to support the applications of CoBrA, a broker-centric agent architecture for building smart meeting rooms, and MoGATU, a peer-to-peer data management for pervasive environments.

1. Introduction

Pervasive computing is a natural extension of the existing computing paradigm. In the pervasive computing vision, computer systems will seamlessly integrate into the life of everyday users, providing them with services and information in an "anywhere, anytime" fashion. We believe in this open and dynamic environment, intelligent computing entities must be able to share knowledge, reason about their environment, and interoperate.

In the past, many prototyping systems and architectures have successfully demonstrated aspects of the pervasive computing paradigm, e.g., handheld devices are augmented with context-aware applications to create personalized tour guides for the museum visitors [1, 18], the user interfaces and applications on a resource-poor mobile device can dynamically migrate to a nearby resource-rich stationary computer when the user enters the office [3], and systems can route telephone calls to the present location of a user by tracking the location of an electronic badge that the user wears [31]. However, they offer only weak support for knowledge sharing and reasoning.

A significant source of this weakness is that they are not built on a foundation of common ontologies with explicit semantic representation [9, 13]. For example, the location information of a user is widely used for guiding the adaptive behavior of the systems [27, 32, 29]. However, none have taken advantageous of the semantics of spatial relations in reasoning about the location context of the users. Additionally, many systems use programming language objects (e.g., Java class objects) to represent the knowledge that the computer systems have about the situational environment. Because these representations require the establishment of a prior low-level implementation agreement between the programs that wish to share information, they cannot facilitate knowledge sharing in an open and dynamic environment.

To address these issues, we believe a shared ontology must be developed for supporting knowledge sharing, context reasoning and interoperability in ubiquitous and pervasive computing systems. By defining such ontology, we can help system developers to reduce their efforts in creating ontologies and to be more focused on the actual system implementations.

In this paper, we describe a shared ontology for supporting pervasive computing applications. This ontology called SOUPA – Standard Ontology for Ubiquitous and Pervasive Applications is expressed using the Web Ontology Language OWL and includes modular component vocabularies to represent intelligent agents with associated beliefs, desires, and intentions, time, space, events, user profiles, actions, and policies for security and privacy.

The rest of this paper is organized as follows. In Section 2, we overview the OWL language and other related ontologies. In Section 3, we describe SOUPA and its ontological structure. In Section 4, we discuss how SOUPA can be extended and used to support two different pervasive computing scenarios. One is in the context-aware smart meeting domain based on CoBrA [8], and the other is in the peer-to-peer data management domain based on MoGATU [25]. Conclusions are given in Section 5.

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

Proceedings of the First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous'04)
0-7695-2208-4/04 $20.00 © 2004 IEEE

2. Background

The SOUPA project begins in November 2003 and is part of an ongoing effort of the Semantic Web in UbiComp Special Interest Group[1], an international group of researchers from academia and industry that is using the OWL language for pervasive computing applications and defining ontology-driven use cases demonstrating aspects of the ubiquitous computing vision.

The goal of the project is to define ontologies for supporting pervasive computing applications. The design of SOUPA is driven by a set of use cases. While the SOUPA vocabularies overlaps with the vocabularies of some existing ontologies, the merits of SOUPA is in providing pervasive computing developers a shared ontology that combines many useful vocabularies from different consensus ontologies. We believe SOUPA can help developers who are inexperience in knowledge representation to quickly begin building ontology-driven applications without needing to define ontologies from scratch, and they can focus more on the functionalities of the actual system implementations.

2.1. The Web Ontology Language OWL

The OWL language is a Semantic Web language for use by computer applications that need to process the content of information instead of just presenting information to humans [20]. This language is developed in part of the Semantic Web initiatives sponsored by the World Wide Web Consortium (W3C).

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 it leaves much to be desired as a language for expressing complex knowledge. Enhancements to the basic XML, such as XML Schemas, 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.

OWL is a knowledge representation 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 [21].

The normative OWL exchange syntax is RDF/XML. Ontologies expressed in OWL are usually placed on web servers as web documents, which can be referenced by other ontologies and downloaded by applications that use ontologies. In this paper, we refer to these web documents as the *ontology documents*.

2.2. Related Ontologies

Part of the SOUPA vocabularies are adopted from a number of different consensus ontologies. The strategy for developing SOUPA is to borrow terms from these ontologies but not to import them directly. Although the semantics for importing ontologies is well defined [2], by choosing not to use this approach we can effectively limit the overhead in requiring reasoning engines to import ontologies that may be irrelevant to pervasive computing applications. However, in order to allow better interoperability between the SOUPA applications and other ontology applications, many borrowed terms in SOUPA are mapped to the foreign ontology terms using the standard OWL ontology mapping constructs (e.g., `owl:equivalentClass` and `owl:equivalentProperty`).

The ontologies that are referenced by SOUPA include the Friend-Of-A-Friend ontology (FOAF) [5, 26], DAML-Time and the entry sub-ontology of time [22], the spatial ontologies in OpenCyc [19], Regional Connection Calculus (RCC) [28], COBRA-ONT [7], MoGATU BDI ontology [23], and the Rei policy ontology [15]. In the rest of this section, we describe the key features of these related ontologies and point out their relevance to pervasive computing applications.

FOAF This ontology allows the expression of personal information and relationships, and is a useful building block for creating information systems that support online communities [12]. Pervasive computing applications can use FOAF ontologies to express and reason about a person's contact profile and social connections to other people in their close vicinity.

DAML-Time & the Entry Sub-ontology of Time The vocabularies of these ontologies are designed for expressing temporal concepts and properties common to any formalization of time. Pervasive computing applications can use these ontologies to share a common representation of time and to reason about the temporal orders of different events.

OpenCyc Spatial Ontologies & RCC The OpenCyc spatial ontologies define a comprehensive set of vocabularies for symbolic representation of space. The ontology of RCC consists of vocabularies for expressing spatial relations for

1 `http://pervasive.semanticweb.org`

Proceedings of the First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous'04)
0-7695-2208-4/04 $20.00 © 2004 IEEE

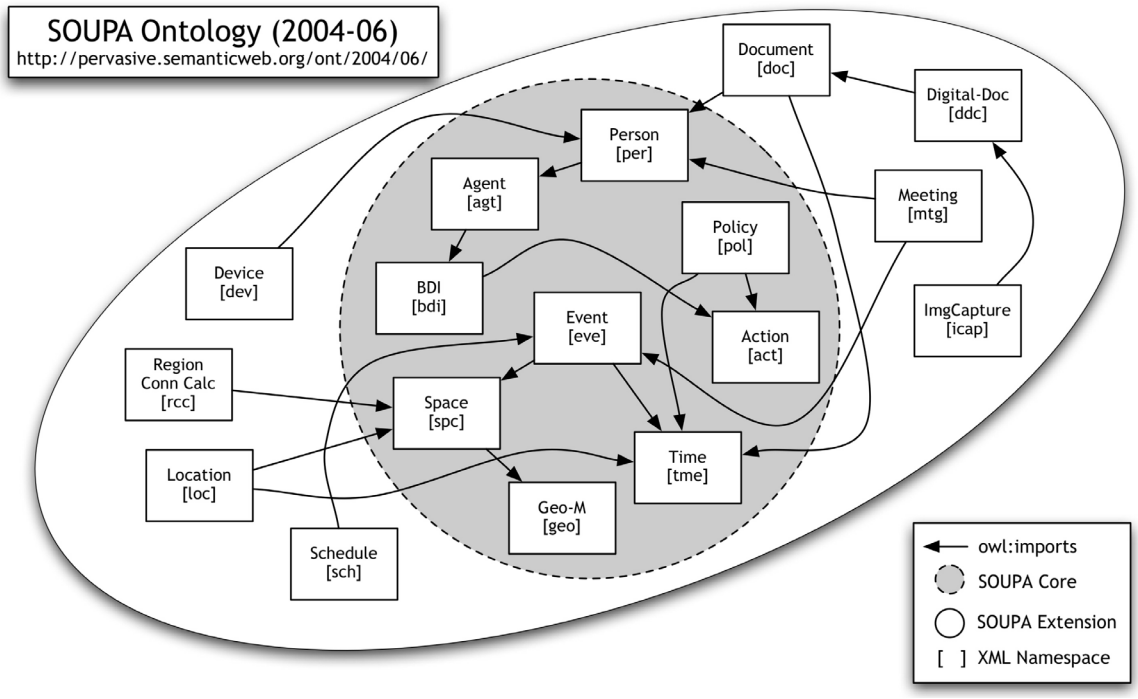

Figure 1. SOUPA consists of two sets of ontology documents: SOUPA Core and SOUPA Extension. The OWL `owl:imports` construct is used to enable a modular design of the ontology. Different domain vocabularies are grouped under different XML namespaces.

qualitative spatial reasoning. In pervasive computing applications, these ontologies can be exploited for describing and reasoning about location and location context [7].

COBRA-ONT & MoGATU BDI Ontology Both COBRA-ONT and MoGATU BDI ontology are aimed for supporting knowledge representation and ontology reasoning in pervasive computing environment. While the design of COBRA-ONT focuses on modeling contexts in smart meeting rooms [7], the design of MoGATU BDI ontology focuses on modeling the belief, desire, and intention of human users and software agents [23].

Rei Policy Ontology The Rei policy language defines a set of deontic concepts (i.e., rights, prohibitions, obligations, and dispensations) for specifying and reasoning about security access control rules. In a pervasive computing environment, users can use this policy ontology to specify high-level rules for granting and revoking the access rights to and from different services [16].

3. SOUPA Ontologies

SOUPA consists of two distinctive but related set of ontologies: SOUPA Core and SOUPA Extension. The set of the SOUPA Core ontologies attempts to define generic vocabularies that are universal for different pervasive computing applications. The set of SOUPA Extension ontologies, extended from the core ontologies, define additional vocabularies for supporting specific types of applications and provide examples for the future ontology extensions.

Note that the structure of the ontology merely suggests certain vocabularies are more general than the others in supporting pervasive computing applications, and there is no inherent computational complexity difference in adopting either set of the ontologies.

3.1. SOUPA Core

This set of ontologies consists of vocabularies for expressing concepts that are associated with person, agent, belief-desire-intention (BDI), action, policy, time, space, and event. The ontologies are grouped into nine distinctive ontology documents. Figure 1 shows a diagram of the ontology documents and their associated relations.

3.1.1. Person This ontology defines typical vocabularies for describing the contact information and the profile of a person. The OWL class `per:Person` is defined to represent a set of all people in the SOUPA domain, and is equiva-

Proceedings of the First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous'04)
0-7695-2208-4/04 $20.00 © 2004 IEEE

lent to the `foaf:Person` class in the FOAF ontology (i.e., the `owl:equivalentClass` property holds between the `per:Person` and `foaf:Person` class). An individual of the class can be described by a set of properties, which include basic profile information (name, gender, age, birth date, etc.), the contact information (email, mailing address, homepage, phone numbers, instant messaging chat ID, etc.), and social and professional profile (people that a person is friend of, organizations that a person belongs to). In addition, all property vocabularies that are applicable to describe a person in the FOAF ontology can also be used to describe an individual of the `per:Person` class. This is because all individuals of the `per:Person` class are also individuals of the `foaf:Person` class. The following shows a partial ontology description of the person Harry Chen:

```
<per:Person>
  <per:firstName
     rdf:datatype="&xsd;string>Harry</per:firstName>
  <per:lastName
     rdf:datatype="&xsd;string>Chen</per:lastName>
  <per:gender rdf:resource="&per;Male"/>
  <per:birthDate
     rdf:datatype="&xsd;date">1976-12-26</per:birthDate>

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

  <per:hasSchoolContact rdf:resource="#SchoolContact"/>
  <per:hasHomeContact rdf:resource="#HomeContact"/>

  <foaf:workplaceHomepage
     rdf:resource="http://ebiquity.umbc.edu"/>
  <foaf:workplaceHomepage
     rdf:resource="http://www.umbc.edu"/>
  <foaf:workplaceHomepage
     rdf:resource="http://www.cs.umbc.edu"/>
</per:Person>

<per:ContactProfile rdf:ID="SchoolContact">
  <per:address rdf:datatype="&xsd;string">
    Dept. of CSEE, UMBC, 1000 Hilltop Circle,
    Baltimore, MD 21250, USA
   </per:address>
   <per:phone
      rdf:datatype="&xsd;string>
      +1-410-455-8648
      </per:phone>
   <per:email
      rdf:resource="mailto:harry.chen@umbc.edu"/>
   <per:im
      rdf:resource="aim:goim?screenname=hc1379"/>
</per:ContactProfile>

<per:Email rdf:about="mailto:harry.chen@umbc.edu"/>
<per:Homepage rdf:about="http://www.aim.com"/>
<per:ChatID rdf:about="aim:goim?screenname=hc1379">
  <per:providedBy rdf:resource="http://www.aim.com"/>
</per:ChatID>

<per:ContactProfile rdf:ID="HomeContact">
  ...
</per:ContactProfile>

<foaf:knows>
  <foaf:Person>
     <foaf:name>Tim Finin</foaf:name>
     <foaf:mbox_sha1sum>
       49953...148d37
     </foaf:mbox_sha1sum>
  </foaf:Person>
</foaf:knows>
</rdf:RDF>
```

3.1.2. Policy & Action Security and privacy are two growing concerns in developing and deploying pervasive computing systems [6, 17, 13]. Policy is an emerging technique for controlling and adjusting the low-level system behaviors by specifying high-level rules [11].

The SOUPA policy ontology defines vocabularies for representing security and privacy policies and a description logic based mechanism for reasoning about the defined policies. The defined vocabularies in this ontology are influenced by the Rei policy language [15].

A policy is a set of rules that is specified by a user or a computing entity to restrict or guide the execution of actions. For example, in the context of system security, a system administrator may use policies to define who has the right to execute what services; in the context privacy protection, a user may use policies to restrict the type of personal information that can be shared by the public services.

The ontology representation of an action is defined in the `action` ontology document. The class `act:Action` represents a set of all actions. Individuals of this class can have a set of property values, which include (i) `act:actor` – the entity that performs the action, (ii) `act:recipient` – the entity that receives the effect after the action is performed, (iii) `act:target` – the object that the action applies to, (iv) `act:location` – the location at where the action is performed, (v) `act:time` – the time at which the action is performed, (vi) `act:instrument` – the thing that the actor uses to perform the action.

The following shows a partial ontology that defines a special class of knowledge sharing action:

```
<owl:Class rdf:ID="ShareHarryLocInfoWithEBMembers">
    <owl:intersectionOf rdf:parseType="Collection">
       <owl:Class rdf:about="&act;Action"/>

       <owl:Restriction>
         <owl:onProperty rdf:resource="&act;actor">
         <owl:hasValue>
            <agt:Agent rdf:about="ctb@cobra1.cs.umbc.edu"/>
         </owl:hasValue>
       </owl:Restriction>

       <owl:Restriction>
          <owl:onProperty rdf:resource="&act;target"/>
          </owl:allvaluesFrom
            rdf:resource="#LocationContextOfHarry"/>
       </owl:Restriction>

       <owl:Restriction>
          <owl:onProperty rdf:resource="&act;recipient"/>
          <owl:allValuesFrom
             rdf:resource="&eb;EbiquityMembers"/>
       </owl:Restriction>
    </owl:intersectionOf>
</owl:Class>

<owl:Class rdf:ID="LocationContextOfHarry">
   <owl:intersectionOf rdf:parseType="Collection">
      <owl:Class rdf:about="&loc;LocationContext"/>
      <owl:Restriction>
         <owl:onProperty
           rdf:resource="&loc:locationContextOf"/>
         <owl:hasValue>
```

```
            <per:Person
              rdf:about="http://umbc.edu/people/hchen4"/>
          </owl:hasValue>
        </Restriction>
    </owl:intersectionOf>
</owl:Class>
```

The above example describes a set of actions of which the actor is the agent `ctb@cobra1.cs.umbc.edu`, the recipient is any member of the eBiquity group, the target, or the information to be shared, is Harry's location information.

In SOUPA, a policy consists of rules that either *permit* or *forbid* the execution of certain described actions. Defined in the `policy` ontology document, the `pol:Policy` class represents a set of all policies. For a given policy individual, it may be associated with one or more `pol:permits` or `pol:forbids` properties. The range of these two properties are the `pol:PermittedAction` class and the `pol:ForbiddenAction` class, respectively.

The following example shows a policy that gives the agent `ctb@cobra1.cs.umbc.edu` the permission to share Harry's location information with all eBiquity members:

```
<pol:Policy rdf:about="&cobra;harrychen-policy">
  <pol:policyOf>
    <per:Person
       rdf:about="http://umbc.edu/people/hchen4">
       <per:name
          rdf:datatype="&xsd;string">Harry Chen</per:name>
    </per:Person>
  </pol:policyOf>

  <pol:defaultPolicyMode
rdf:resource="&pol;RequiresExplicitPermission"/>

  <pol:permits
    rdf:resource="#ShareHarryLocInfoWithEBMembers"/>
</pol:Policy>
```

The policy ontology also defines vocabularies for describing meta information about individual policies. This information includes the author of a policy (`pol:creator`), the entity that enforces a policy (`pol:enforcer`), the creation time of a policy (`pol:createdOn`), and the default reasoning mode of a policy (`pol:defaultPolicyMode`).

The design of the SOUPA policy exploits *classification* as a means to reason about policies. A typical process flow of the system implementation is the following: (i) a user or a system administrator defines a policy, (ii) the policy is transmitted to the appropriate policy enforcer (e.g, a security or a privacy protection agent), (iii) before the policy enforcer can permit other agents to perform an action, it creates an explicit representation of the action using the SOUPA action ontology, (iv) this represented action is then loaded into a description logic reasoner (e.g., Racer [14] or FaCT[2]) along with the associated ontology, (v) the policy enforcer will permit the execution of the action if the action is classified as type of `pol:PermittedAction`, and it will forbid the execution of the action if the action is classified as type of `pol:ForbiddenAction`.

In case if an input action is classified as both `pol:PermittedAction` and `pol:ForbiddenAction`, then the policy enforcer will report there is an inconsistency in the policy and may forbid the execution of the action by default. In case if the action cannot be classified as either classes, the policy enforcer will decide whether the action should be permitted or forbidden based on the default policy mode (see the above example). If the mode is `pol:RequiresExplicitPermission`, then the action will be forbidden. If the mode is `pol:RequiresNoExplicitPermission`, then the action will be permitted.

3.1.3. Agent & BDI When building intelligent pervasive computing systems, sometimes it is useful to model computing entities as *agents* [33]. In SOUPA, agents are defined with a strong notion of agency [33], which is characterized by a set of *mentalistic* notions such as knowledge, belief, intention, and obligation. In this ontology, both computational entities and human users can be modeled as agents.

When the goals, plans, desires, and beliefs of different agents are explicitly represented in the ontologies, this information can help independently developed agents to share a common understanding of their "mental" states, helping them to cooperate and collaborate. The explicitly represented human user's mental states can help computing agents to reason about the specific needs of the users in a pervasive environment.

Two ontology documents are related to this ontology: `agent` and `bdi`. The `agt:Agent` class represents a set of all agents in the SOUPA domain and is associated with three properties that can be used to characterize an agent's "mental" state: `agt:believes`, `agt:desires`, and `agt:intends`. The respective range values of these properties are the `bdi:Fact`, `bdi:Desire`, and `bdi:Intention` classes. The goals of an agent are considered to be a special type of desire, which is expressed by defining the `agt:hasGoal` property as a sub-property of the `agt:desires` property.

The `bdi:Fact` class is a subclass of the `rdf:Statement` class, which represents a set of reified RDF statements [4]. A reified RDF statement consists of the `rdf:subject`, `rdf:object`, and `rdf:predicate` properties.

The `bdi:Desire` class defines a set of world states that agents desire to bring about. Every instances of this class can be characterized by the property `bdi:endState`. The range restriction of this property is unspecified in the `bdi` ontology document. Application developers are responsible for defining the representation of different world states.

2 `http://www.cs.man.ac.uk/~horrocks/FaCT/`

Some suggested representations are (i) symbolic names, e.g., a set of pre-defined RDF resource URI and (ii) meta-representation, e.g., each world state description is a set of reified RDF statements.

The bdi:Intention class represents a set of plans that agents intend to execute. Plans are defined in terms of actions, pre-conditions, and effects. The bdi:Plan class is defined as a subclass of the act:Action class with additional properties, namely bdi:preCondition and bdi:effect.The representation of pre-conditions and effects are unspecified in this ontology, and it is left to be defined by the application ontologies.

Sometimes it may be useful to describe whether or not different desires of an agent are in conflict of each other, and whether or not certain desires are achievable. The cause of desire conflicts may be due to inconsistent beliefs in the knowledge base or conflicting user preferences or systems policies. The cause of unachievable desires may be due to the change of situational conditions. In the bdi ontology document, different subclasses of the bdi:Desire class, bdi:ConflictingDesire, bdi:Non-ConflictingDesire, bdi:AchievableDesire, and bdi:UnachievableDesire, are defined for classifying different types of agent desires.

3.1.4. Time SOUPA defines a set of ontologies for expressing time and temporal relations. They can be used to describe the temporal properties of different events that occur in the physical world.

Part of the SOUPA ontology adopts the vocabularies of the DAML-time and the entry sub-ontology of time. The basic representation of time consists of the tme:TimeInstant and tme:TimeInterval classes. All individual members of these two classes are also members of the tme:TemporalEntity class, which is an OWL class that is defined by taking the union of the tme:TimeInstant and tme:TimeInterval classes. The set of all temporal things that are divided into two disjoint classes: tme:InstantThing, things with temporal descriptions that are type of time instant, and tme:IntervalThing, things with temporal descriptions that are type of time interval. The union of these two classes forms the tme:TemporalThing class.

In order to associate temporal things with date/time values (i.e., their temporal descriptions), the tme:at property is defined to associate an instance of the tme:InstantThing with an XML xsd:dateTime datatype value (e.g., 2004-12-25T12:32:12), and the tme:from and tme:to properties are defined to associate an instance of the IntervalThing with two different tme:TimeInstant individuals. The following example shows the representation of a time interval with the associated temporal description:

```
<tme:TimeInterval>
  <tme:from>
    <tme:TimeInstant>
      <tme:at rdf:datatype="xsd;dateTime">
        2004-02-01T12:01:01
      </tme:at>
    </tme:TimeInstant>
  </tme:from>
  <tme:to>
    <tme:TimeInstant>
      <tme:at rdf:datatype="xsd;dateTime">
        2004-02-11T13:41:21
      </tme:at>
    </tme:TimeInstant>
  </tme:to>
</tme:TimeInterval>
```

For describing the order relations between two different time instants, the ontology defines the following properties: tme:before, tme:after, tme:beforeOrAt, tme:afterOrAt, and tme:sameTimeAs. Both tme:before and tme:after properties are defined of type owl:TransitiveProperty. The tme:sameTimeAs property expresses that two different time instants are associated with equivalent date/time values and is defined of type owl:SymmetricProperty.

For describing the order relations between two different temporal things (i.e., time instants and time intervals), the ontology defines the following properties: tme:startsSoonerThan, tme:startsLaterThan, tme:startsSameTimeAs, tme:endsSoonerThan, tme:endsLaterThan, tme:endsSameTimeAs, tme:startsAfterEndOf, and tme:endsBeforeStartOf. The first three properties respectively express that for any two given temporal things A and B, the starting time of A is before the starting time of B, the starting time of A is after the starting time of B, and the starting time of A is the same as the starting time of B. The next three properties respectively express that for any two given temporal things A and B, the ending time of A is before the ending time of B, the ending time of A is after the ending time of B, and the ending time of A is the same as the ending time of B. The tme:startsAfterEndOf property expresses that the beginning of one temporal thing is after the ending of another temporal thing, and the tme:endsBeforeStartOf property expresses the inverse of this property.

In the future we plan to extend this ontology to adopt or map to additional vocabularies from the RDF Calendar ontologies[3] for modeling time intervals that may contain repeating time intervals or instants. This new feature will be useful for representing recurrent events such as weekly meetings and classes.

3.1.5. Space This ontology is designed to support reasoning about the spatial relations between various types of geographical regions, mapping from the geo-spatial coordinates to the symbolic representation of space and *vice versa*, and

3 http://www.w3.org/2002/12/cal/

the representation of geographical measurements of space. Part of this ontology vocabularies are adopted from the spatial ontology in OpenCyc and the OpenGIS vocabularies [10].

Two ontology documents are related to this ontology: `space` and `geo-measurement`. The first ontology document defines a symbolic representation of space and spatial relations, and the second document defines typical geospatial vocabularies (e.g., longitude, latitude, altitude, distance, and surface area).

In the symbolic representation model, the `spc:SpatialThing` class represents a set of all things that have spatial extensions in the SOUPA domain. All spatial things that are 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 that is controlled by some political body (e.g., the country 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. For example, a system may apply different sets of privacy protection schemes based on the policies defined by the local political entities.

To support spatial containment reasoning, individual members of the `spec:GeographicalSpace` class can relate to each other through the `spc:spatiallySubsumes` and `spc:spatiallySubsumedBy` properties. For example, a country region may spatially subsumes a state region, a state region may spatially subsumes a building, and a building may spatially subsumes a room. Knowing the room in which a device is located, we can infer the building, the state and the country that spatially subsumes the room.

In the geo-spatial representation model, the individual members of the `spc:SpatialThing` class are described by location coordinates (i.e., longitude, latitude, and altitude). This relation is expressed by the `spc:hasCoordinates` property, the range of which is the `geo:LocationCoordinates` class. In this model, multiple location coordinates can be mapped to a single geographical region (e.g., a university campus typically covers multiple location coordinates.). This relation is useful for defining spatial mapping between different geographical locations and GPS coordinates. This information can enable a GPS-enabled device to query the symbolic representation of its present location for a given set of longitude, latitude, and altitude.

3.1.6. Event Events are event activities that have both spatial and temporal extensions. An event ontology can be used to describe the occurrence of different activities, schedules, and sensing events. In the `event` ontology document, the `eve:Event` class represents a set of all events in the domain. However, the definition of this class is silent about its temporal and spatial properties.

The `eve:SpatialTemporalThing` class represents a set of things that have both spatial and temporal extensions, and it is defined as the intersection of the `tme:TemporalThing` and `spc:SpatialThing` classes. To specifically describe events that have both temporal and spatial extensions, `eve:SpatialTemporalEvent` class is defined as the intersection of the `eve:SpatialTemporalThing` and `eve:Event` classes.

The following example shows how the ontology can be used to describe an event in which a Bluetooth device has been detected on 2004-02-01 at 12:01:01 UTC, and the event occurs at a location that is described by longitude -76.7113 and latitude 39.2524:

```
<owl:Class rdf:ID="DetectedBluetoothDev">
  <rdfs:subClassOf
   rdf:resource="&eve;TemporalSpatialEvent"/>
</owl:Class>

<owl:ObjectProperty rdf:ID="foundDevice">
  <rdfs:domain
  rdf:resource="#DetectedBluetoothDev"/>
</owl:ObjectProperty>

<DetectedBluetoothDev>
  <spc:hasCoordinates>
    <geo:LocationCoordinates>
      <geo:longitude rdf:datatype...>
        -76.7113
      </geo:longitude>
      <geom:latitude rdf:datatype...>
        39.2524
      </geom:latitude>
    </geo:LocationCoordinates>
  </spc:hasCoordinates>

  <foundDevice rdf:resource="url-x-some-device"/>
  <tme:at>
    <tme:TimeInstant>
      <tme:at rdf:datatype="xsd;dateTime">
        2004-02-01T12:01:01
      </tme:at>
    </tme:TimeInstant>
  </tme:at>
<DetectedBluetoothDev>
```

3.2. SOUPA Extension

The SOUPA Extension ontologies are defined with two purposes: (i) define an extended set of vocabularies for supporting specific types pervasive application domains, and (ii) demonstrate how to define new ontologies by extending the SOUPA Core ontologies. At present, the SOUPA Extension consists of experimental ontologies for supporting per-

vasive context-aware applications in smart spaces and peer-to-peer data management in a pervasive computing environment. Due to space limitation, in this section, we briefly describe the existing SOUPA Extension ontologies.

Meeting & Schedule For describing typical information associated with meetings, event schedules, and event participants.

Document & Digital Document For describing meta-information about documents and digital documents, e.g., the creation date and the author of a document, the source URL of a digital document, file size, and file type.

Image Capture When a camera phone takes a picture, this event is type of image capturing event. This ontology defines vocabularies for describing image capturing events (where and when a picture is taken, which device has taken the picture, etc.).

Region Connection Calculus A spatial ontology that supplements the core space ontology. Based on the Region Connection Calculus [28], this ontology defines vocabularies for expressing spatial relations for qualitative spatial reasoning.

Location For describing sensed location context of a person or an object. The location context is information that describes the whereabouts of a person or an object, which includes both temporal and spatial properties.

4. SOUPA Applications

To demonstrate SOUPA is feasible for supporting pervasive computing applications, we are prototyping two use case scenarios. One is in the pervasive context-aware meeting room domain based on CoBrA, and the other is in the peer-to-peer data management domain based on MoGATU. Our objective is to show that by defining a shared pervasive computing ontology, we can help system developers to reduce their efforts in creating ontologies and to be more focused on the actual implementation of their systems.

4.1. CoBrA

CoBrA is a broker-centric agent architecture for supporting context-aware systems in smart spaces [8]. Central to the architecture is the presence of a Context Broker, an intelligent agent that runs on a resource-rich stationary computer in the space. It's responsible for acquiring and maintaining context knowledge, reasoning about the information that cannot be directly acquired from sensors (e.g., intentions, roles, temporal and spatial relations), detecting and resolving inconsistent knowledge that is stored in the shared model of context, and protecting user privacy by enforcing policies. CoBrA has been used to prototype a smart meeting system called EasyMeeting [9], in which the Context Broker uses ontologies and logic inference rules to reason about the location of meeting attendees based on the location of their personal Bluetooth cellphones, and share this information with other meeting services in the conference room.

The SOUPA ontologies can be used in CoBrA to facilitate knowledge sharing and ontology reasoning. The following use case describes typical uses of the ontologies in CoBrA:

Room 338 is a smart meeting room. On March 8th, 2004, a presentation is scheduled to take place from 1:00-2:30 PM in this room. Moments before the event starts, the room's Context Broker acquires the meeting's schedule from the Web, which is expressed in the `schedule` ontology, and concludes the meeting is about to take place in the Room 338. As the meeting participants begin to arrive, the room's Bluetooth Sensing Agent detects the presences of different Bluetooth enabled devices (e.g., cellphones, PDA's). Because each device has a unique device owner profile, which is expressed in the `person` ontology, the sensing agent can share this information with the Context Broker.

Based on the acquired information (e.g., what type of device it is, and who owns what devices) and without knowing any evidence to the contrary, the Context Broker concludes the owners of the detected devices are also located in the Room 338. Among the arrived participants, Harry the speaker and President Hrabowski the distinguished audience are two people that are listed in the meeting schedule. This information is expressed in the `meeting` and the `schedule` ontologies. The Context Broker shares their location information, which is expressed in the `space` and `time` ontologies, with the subscribed Meeting Management Agent.

Knowing that President Hrabowski has a distinguished audience role, the Meeting Management Agent invokes a greeting service to greet him. At 1:00 PM, the Context Broker informs the the same agent that all listed *key* participants have arrived and that the presentation can start. Knowing all the lights in the meeting are currently switched on and the background music is also playing, the agent tells the light controller to dim the lights and the music service to stop the music.

4.2. MoGATU

MoGATU is a framework for handling pro-active peer-to-peer semantic data management in a pervasive computing environment [24, 25]. The framework treats all devices present in the environment as equal semi-autonomous peers. To provide uniformed communication functionality and to handle data management issues, this framework abstracts

all devices in the environment as a collection of Information Managers, Information Providers, and Information Consumers. It specifies several communication interfaces for supporting ad-hoc IEEE 802.11 and Bluetooth like networks. Each InforMa is responsible for maintaining information about peers in its vicinity. This information includes the types of devices, and information and services they provide. An InforMa also maintains a data cache for storing information obtained from other mobile devices and to cache information generated by its local Providers. Additionally, each InforMa may include a user's profile reflecting some of user's beliefs, desires, and intentions, a model adopted by the SOUPA ontology. MoGATU is targeted toward pervasive environments, which invert the traditional sense of data management in distributed databases that is based on "passive data" and "active users" concepts [30].

A key use of the SOUPA ontologies in MoGATU is to express beliefs, preferences, intentions, and desires of an agent or a user, and the priority values of plans and goals. The following use case describes typical uses of the ontologies in MoGATU:

While Bob is getting ready to leave his new office, his phone rings. It is his new friend Jane asking him to meet her at the local shopping mall. Bob agrees to meet her and notifies his palmtop about the decision. While he is walking through the building toward the parking lot and ultimately toward his car, the palmtop fetches appropriate directions from the web through the office network infrastructure. Once inside a car, the palmtop speaks out the directions and Bob follows them; however, the device knows that Bob has a time constraint when he should meet Jane. It actively queries cars around Bob, to ask them if there is any traffic delay on the selected route, and if they know of an alternate, faster route. It turns out that there is a traffic jam building up on the route, that was suggested by the web service and that also would be planned by Bob's car navigational system. The palmtop, therefore, queries surrounding cars and is able to return with an alternate set of directions that circumvent the afternoon traffic jam. Bob takes the different roads and is able to arrive at the mall's entrance thirty minutes early.

Bob decides to use the extra time to check out local stores. His palmtop learns from the shopping mall broker that it is now located inside the mall. The palmtop proactively evaluates all intentions and desires of Bob, stored in his profile, to see if it can assist Bob in satisfying some of them. It infers that Bob needs a new pair of shoes. Accordingly, it starts to query available shoe stores for their current offers and also tries to negotiate addidional deals. As Bob walks through the mall, the palmtop is able to collect enough evidence store advertisements and other customers' opinions on stores and their offerings. The palmtop combines the collected knowledge and suggests Bob to visit the shoe store in Nordstrom. This is because the palmtop was able to negotiate a 20% sale coupon on Bob's favorite shoe brand. Bob decides to use the offer and purchases a pair of shoes. The palmtop learns that from Bob's credit card statement and removes the associated goal. Consequently, upon next visit of the mall, the palmtop will no longer place a high priority on obtaining deals from shoe stores.

5. Conclusions

The use of ontologies is a key requirement for realizing the vision of pervasive computing. We believe by defining a shared pervasive computing ontology, we can help system developers to reduce their efforts in creating ontologies and to be more focused on the actual system implementations. The SOUPA project is a step towards the standardization of a shared ontology for ubiquitous and pervasive computing applications.

We believe SOUPA will evolve as the pervasive computing research progresses. The use of SOUPA shows great promises in facilitating knowledge sharing and ontology reasoning in both CoBrA and MoGATU. We expect the vocabularies and the structures of SOUPA to change when the research community becomes more experienced in developing ontology-driven applications.

The overall experience in developing the SOUPA ontology was challenging. First, when building a shared ontology that is aimed to reuse a number of different ontologies, it was difficult to decide what is the most appropriate ontology structure for organizing the defined vocabularies. Second, because not all of the reused ontologies were developed using the same ontology language, and some of which were developed to support different types of logic inferences, often it was necessary to modify the structures and the constructs of the existing ontologies before including them into the SOUPA ontology. Third, developing methodologies to measure the success of the SOUPA ontology was difficult.

In the current SOUPA ontology development process, we have taken the following approaches to address the above three issues: (i) we have developed use case scenarios with functional specifications to help us decide the appropriate structures for organizing ontological vocabularies, (ii) we have chosen the OWL DL sub-language as the language to build a shared ontology from the existing ontologies, (iii) we plan to show the usefulness of the SOUPA ontology by prototyping different pervasive computing systems to use SOUPA.

6. Acknowledgments

The authors would like to thank the members of the Semantic Web in UbiComp Special Interest Group for their

critical comments and feedbacks during the SOUPA development.

References

[1] G. D. Abowd, C. G. Atkeson, J. Hong, S. Long, R. Kooper, and M. Pinkerton. Cyberguide: A mobile context-aware tour guide. *ACM Wireless Networks*, pages 421–433, 1997.

[2] S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein. *OWL Web Ontology Language Reference*, w3c recommendation 10 february 2004 edition, February 2004.

[3] F. Bennett, T. Richardson, and A. Harter. Teleporting - making applications mobile. In *Proceedings of 1994 Workshop on Mobile Computing Systems and Applications*, Santa Cruz, December 1994.

[4] D. Brickley and R. Guha. RDF Vocabulary Description Language 1.0: RDF Schema. In *W3C Recommendation*. RDF Core Working Group, 2004.

[5] D. Brickley and L. Miller. FOAF vocabulary specification. In *RDFWeb Namespace Document*. RDFWeb, `xmlns.com`, 2003.

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

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

[8] H. Chen, T. Finin, and A. Joshi. A context broker for building smart meeting rooms. In *Proceedings of the Knowledge Representation and Ontology for Autonomous Systems Symposium, 2004 AAAI Spring Symposium*. AAAI, March 2004.

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

[10] S. Cox, P. Daisey, R. Lake, C. Portele, and A. Whiteside. Geography Markup Language (GML 3.0). In *OpenGIS Documents*. OpenGIS Consortium, 2003.

[11] N. Damianou, N. Dulay, E. Lupu, and M. Sloman. The Ponder Policy Specification Language. *Lecture Notes in Computer Science*, 1995:18–??, 2001.

[12] E. Dumbill. Finding friends with XML and RDF. In *IBM developerWorks, XML Watch*. xmlhack.com, June 2002.

[13] F. L. Gandon and N. M. Sadeh. Semantic web technologies to reconcile privacy and context awareness. *Web Semantics Journal*, 1(3), 2004.

[14] V. Haarslev and R. Möller. Racer system description. In *Proceedings of the International Joint Conference on Automated Reasoning 2001*, 2001.

[15] L. Kagal, T. Finin, and A. Joshi. A Policy Based Approach to Security for the Semantic Web. In *2nd International Semantic Web Conference (ISWC2003)*, September 2003.

[16] L. Kagal, M. Paolucci, N. Srinivasan, G. Denker, T. Finin, and K. Sycara. Authorization and privacy for semantic web services. *AAAI 2004 Spring Symposium on Semantic Web Services*, March 2004.

[17] L. Kagal, J. Parker, H. Chen, A. Joshi, and T. Finin. *Handbook of Mobile Computing*, chapter Security, Trust and Privacy in Mobile Computing Environments. CRC Press, 2004.

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

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

[20] D. L. McGuinness and F. van Harmelen. OWL Web Ontology Language Overview. In *Proposed Recommendation (PR) for OWL*. Web Ontology Working Group, W3C, 2003.

[21] N. F. Noy and D. L. McGuinness. Ontology development 101: A guide to creating your first ontology. Technical Report KSL-01-05, Stanford Knowledge Systems Laboratory, 2001.

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

[23] F. Perich. MoGATU BDI Ontology, 2004.

[24] F. Perich, A. Joshi, T. Finin, and Y. Yesha. On Data Management in Pervasive Computing Environments. *IEEE Transactions on Knowledge and Data Engineering*, October 2003.

[25] F. Perich, A. Joshi, Y. Yesha, and T. Finin. Neighborhood-Consistent Transaction Management for Pervasive Computing Environments. In *14th International Conference on Database and Expert Systems Applications (DEXA 2003)*, Prague, Czech Republic, September 2003.

[26] S. Powers. *Practical RDF*. O'Reilly & Associates, 2003.

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

[28] D. A. Randell, Z. Cui, and A. Cohn. A spatial logic based on regions and connection. In B. Nebel, C. Rich, and W. Swartout, editors, *KR'92. Principles of Knowledge Representation and Reasoning: Proceedings of the Third International Conference*, pages 165–176. Morgan Kaufmann, San Mateo, California, 1992.

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

[30] M. Stonebraker. Position Paper on Monitoring Applications. In *NSF Workshop on Context-Aware Mobile Database Management (CAMM)*, January 2002.

[31] R. Want, A. Hopper, V. Falcão, and J. Gibbons. The active badge location system. Technical Report 92.1, Olivetti Research Ltd. (ORL), 24a Trumpington Street, Cambridge CB2 1QA, 1992.

[32] R. Want, B. Schilit, N. Adams, R. Gold, K. Petersen, D. Goldberg, J. R. Ellis, and M. Weiser. An overview of the PARCTAB ubiquitous computing experiment. *IEEE Personal Communications*, 2(6):28–33, Dec 1995.

[33] M. J. Wooldridge and N. R. Jennings. Intelligent agents: Theory and practice. *Knowledge Engineering Review*, 10(2):115–152, June 1995.