Toward a Domain-Independent Semantic Model for Context-Aware Computing

Renato de Freitas Bulcão Neto, Maria da Graça Campos Pimentel
Institute of Mathematical Sciences and Computing
University of São Paulo, Brazil
{rbulcao, mgp}@icmc.usp.br

Abstract

Context-aware systems assist people's everyday tasks by adapting their behavior based on context information gathered from instrumented environments. Context is any relevant information describing entities in a user-computer interaction. A pertinent issue to developing such systems is how to represent context. High-level context models prevent the development of ad hoc and limited schemes for context management. Moreover, the more formal a context model is, the better is the ability for reasoning about context. This paper presents our work toward a domain-independent ontology-based context model, which provides a set of general classes, properties, and relations so lower ontologies can import them for particular domains. We show how this context model can be extended in a ubiquitous e-learning scenario on campus. Built upon our model, we also present a configurable service infrastructure for the prototyping of semantic context-aware systems.

1. Introduction

Ubiquitous computing[1] has emerged as a paradigm where users interact with instrumented environments that transparently embed applications. The main idea is to support users in everyday tasks carried out away from the desktop [1]. One of the research themes in ubicomp is that of context-aware computing, where applications customize their behavior based on context sensed from those instrumented environments [2]. Context is "any relevant information about a user-application interaction, including the user and the application themselves" [3]. For instance, using sensors network and computers, an elder care system infers whether and how people with early-stage cognitive decline perform activities of daily living [4].

The development of context-aware systems is usually a complex and time-consuming task for developers [5][6]. A relevant issue for the development of these systems is how to represent context. High-level context models avoid the development of ad hoc and limited schemes for the management of context. Context models also need representation languages that use broadly accepted standards so as to make it easier the sharing and the reuse of context between context-aware systems [7]. Furthermore, the more formal a context model is, the better is the ability for context-aware systems to infer from context information.

Much research on context modeling has been done in the past few years. The early context models represent context in an ad hoc manner for particular applications, which include the Xerox ParcTab [8] and the Olivetti's Active Badge [9]. The Cooltown project represents each real world object — people, places and devices — with an HTML web page, which is an approach that lacks formality and expressiveness regarding context modeling [10]. Most recent models focus on XML representations with no semantics associated such as the *Context Specification Language* [11] and the *MediaObject* context model [12]. More formal context models exploit conceptual data modeling techniques such as those provided by Entity-Relationship models [13] and UML diagrams [14]. Artificial intelligence approaches have been also used for context modeling such as first order predicate calculus [15] and ontologies [5] [7]. We advocate that a better support for context reuse, sharing and reasoning is achieved through Semantic Web ontologies [16] as underlying formal model.

In this paper, we propose a domain-independent ontological model for representing context that provides a set of general classes, properties, relations, and restrictions so that lower ontologies can import them for particular domains. The design space for building our context model derives from dimensions for context modeling debated in the literature [2] [17]: *who*, *where*, *when*, *what* and *how*. Regarding our context model, those contextual dimensions represent the basic concepts of actors, location, time, activities, and devices respectively, as well as the relationships between these concepts. We have followed the Stanford's methodology [18] as ontology development methodology to create

[1]The term "ubiquitous computing" is hereafter referred to "ubicomp".

the semantic context model. Concepts of well known Semantic Web ontologies have been reused so as to assist in the building of our context model such as FOAF [19], OpenCyc [20] and CC/PP [21]. We also show how this ontological context model can be extended by a lower ontology built for a ubicomp e-learning scenario on campus.

Based on our semantic context model, we have implemented the service infrastructure for context management *Semantic Context Kernel*, which includes configurable services for storage, query and inference over semantic context, and a discovery service. For instance, the context persistence service allows application developers to choose the type of persistent storage and the type of RDF [22] serialization for semantic context. The main feature of the Semantic Context Kernel relies on the configurability of its services according to context-aware systems' requirements in order to facilitate the prototyping of such systems.

Section 2 outlines the design space of context-aware applications. In Section 3 we present our ontology-based context model. Section 4 describes the architecture of the Semantic Context Kernel. Section 5 reviews related work. In Section 6 we present concluding remarks and future work.

2. Design Space of Context-Aware Applications

Most of context-aware applications have exploited identity and location information to provide users with valuable services. However, context is more than identity and location. The literature has largely discussed semantic dimensions for context modeling in ubicomp interactions. In our work, we have exploited five semantic dimensions (the so-called *W4H*): identity (*who*), location (*where*), time (*when*), activity (*what*) [2] and devices profiles (*how*) [17]. Although these semantic dimensions do not suggest completeness, we advocate that they form a set of issues to be addressed when designing context-aware systems:

- *Who* are the interaction's participants?
- *Where* does the interaction take place?
- *When* does the interaction take place?
- *What* does the interaction describe?
- *How* is context captured and accessed in the interaction?

The *who* component of a context-aware application deals with actors, i.e., entities that perform some action (e.g. people). Besides identifying the *types* of actors, systems designers should also design applications to take advantage of actor's *roles* (e.g. lecturers and participants in a conference system). The *relation* between actors should also be considered on the design of context-aware systems so as to represent the social connections between them (e.g. the parenthood relation between people in a home domain). Finally, the *artifacts* produced by actors can be an important issue to be addressed as in the design of document-oriented systems (e.g. the creators and editors of documents).

The *where* dimension addresses *physical locations* involved in a ubicomp interaction. It should not only deal with interactions occurring in a single location, but also those occurring in different places. If a system designer understands where interactions may take place, he can obtain valuable information to design the system architecture (e.g. considering distribution of context). Hence *mobility* is a key issue on the design of context-aware applications (e.g. mobility of people and devices).

The *when* component of a context-aware application can be used to *temporally situate* an event. A simple approach for situating events is registering everything. However, the system designer should be aware of which events are relevant so that they can be time-indexed. In addition, the frequency of relevant events is an important issue to be addressed because events may occur on regular intervals or not. These issues may be key points on the design of a service infrastructure such as an event notification service. The *when* dimension may also allow the *prediction* of a baseline of behavior if temporal information is combined with other dimensions. For instance, a system might notice whether an elderly person does not follow his routine [4].

The *what* dimension describes *activities* performed in a ubicomp interaction. Although this is a domain-dependent information, it is hard to obtain. One solution is to combine information from other dimensions to infer activity information [5]: a system can infer that "Steve (*who*) is attending a meeting (*what*) because he is at a meeting room (*where*) with his supervisor (*who*)". Three issues should be considered by system designers when modeling activity information: (i) he must fully understand the activities that he aims to support; (ii) he must choose the suitable types of dimensions to be combined; (iii) despite the likely inconsistence of sensor-based context, the more related and relevant context is combined, the more accurate is the inferred value.

The *how* component of a context-aware application describes physical and functional characteristics of *interaction devices* [17]. This is a relevant issue because ubicomp interactions are characterized by *devices heterogeneity*. Most devices have physical constraints in terms of hardware (e.g. screen size and I/O capabilities) and software (e.g. audio and video encoders supported). Both the *profile* and the *mobility* of devices have importance to system designers when personalized information, services, or user interfaces [23] are central issues.

The next section presents our context model based on Semantic Web ontologies, and how it can be employed in an e-learning scenario on campus.

3. A Semantic Context Model

Several efforts have been developed toward the Semantic Web vision such as standards for syntactic, structural, and semantic representation, metadata architectures, and ontology languages. We chose the OWL (*Ontology Web Language*) [24] for context modeling due to several reasons. First, it is a W3C recommendation that employs web standards for information representation such as RDF [22] and XML Schema [25]. Second, it allows the necessary semantic interoperability between context-aware systems. Finally, it provides a high degree of inference making by providing additional vocabulary along with a formal semantics to define classes, properties, relations and axioms.

Figure 1 depicts an overview of our ontological context model. Based on the *W4H* dimensions described in Section 2, the context model represents the basic concepts of actors, location, time, activities, and devices as well as the relations between these concepts.

In order to build our context model, we have followed the Noy and McGuinness [18] ontology 101 methodology since it is relatively simple to learn and use. Regarding its step 1 (*Determine the domain and scope of the ontology*), the competency questions that our domain-independent model should be able to answer must be general in scope as those shown in Section 2. Regarding the step 2 (*Consider reusing existing ontologies*), concepts of Semantic Web vocabularies have been borrowed in order to address every dimension as well as to serve as guidance for context modeling.

Next, we describe our ontological context model in terms of both each related semantic dimension and reused vocabularies, as well as in terms of modeling characteristics and reasoning support.

3.1 Who-based ontologies

The core of this set of ontologies is the *Actor* ontology. It models the profile of all entities that can perform actions in a ubicomp environment such as people, groups and organizations. According to **Figure** 1, the *Actor* ontology imports ontologies that we have built to deal with actors' profile that we describe as follows.

The *role* ontology describes the actors' social role in the real world (e.g. employee, manager and student). The *contact* ontology represents the different types of actors' contact information, including online accounts (e.g. email and instant messaging chat ID), phone numbers (e.g. mobile and workplace), contact addresses (e.g. home and workplace). The *expertise* ontology models knowledge areas, for instance, to make it possible to describe and interrelate people's knowledge, or to find someone with a particular expertise or topic of interest. The *relationship* ontology models social relationships between people, for instance, to manage their friends and colleagues networks. The *project* ontology describes meta-information associated to projects and the links with actors (e.g. membership). Finally, the *document* ontology models documents made by actors (e.g. web pages) by means of meta-information (e.g. subject).

The FOAF (*Friend of a Friend*) vocabulary [19] has contributed with concepts that model basic characteristics of actors and the supporting ontologies, which also reuse concepts of consensus vocabularies such as Dublin Core [26] and vCard [27]. Example 1 describes a simple profile of a person whose name is "Steve Orr", his birthday, instant messaging chat ID (*ICQ number 10042345*), and someone with whom he works identified by "IanBattle".

```
<!--            Example 1            -->
<act:Person rdf:ID="SteveOrr">
  <act:hasName>Steve Orr</act:hasName>
  <act:hasBirthday>07/12/1965</act:hasBirthday>
  <act:hasContactProfile rdf:resource="#SteveIM"/>
  <rel:worksWith rdf:resource="#IanBattle"/>
</act:Person>
<inf:IMAccount rdf:ID="SteveIM">
  <inf:imType>ICQ</inf:imType>
  <inf:imValue>10042345</inf:imValue>
</inf:IMAccount>
```

From the OWL's point of view:

- *act:Person* and *inf:IMAccount* are modeled as *owl:Class*. The former is subclass of *act:Actor*, whereas the latter is subclass of *inf:OnlineAccount*, which in turn is subclass of *inf:ContactProfile*;
- *act:hasContactProfile* is an *owl:ObjectProperty*; its domain is *act:Person* and the range is *inf:ContactProfile*;
- *act:hasName* and *act:hasBirthday* are represented as *owl:DatatypeProperty*. The domain and the range of both classes are the class *act:Person* and the XML Schema string datatype, respectively;
- *inf:imValue* and *inf:imType* are *owl:DatatypeProperty*. The domain of both classes is the class *act:IMAccount*. The range of *inf:imValue* is the XML Schema string datatype, whereas the range of *inf:imType* is an enumerated datatype, i.e., a list of possible values for this property (e.g. ICQ, Yahoo, and MSN);
- *rel:worksWith* is described as *owl:SymmetricProperty*. Hence its domain and range are the same: the class *act:Person*. For this kind of property, an inference engine deducts that if "SteveOrr" *worksWith* "IanBattle" then this relation is reciprocal.

3.2 Where-based ontology

The *Location* ontology aims to describe the whereabouts of real world entities. Location information is not only related to usual information (e.g. street, city and room), but

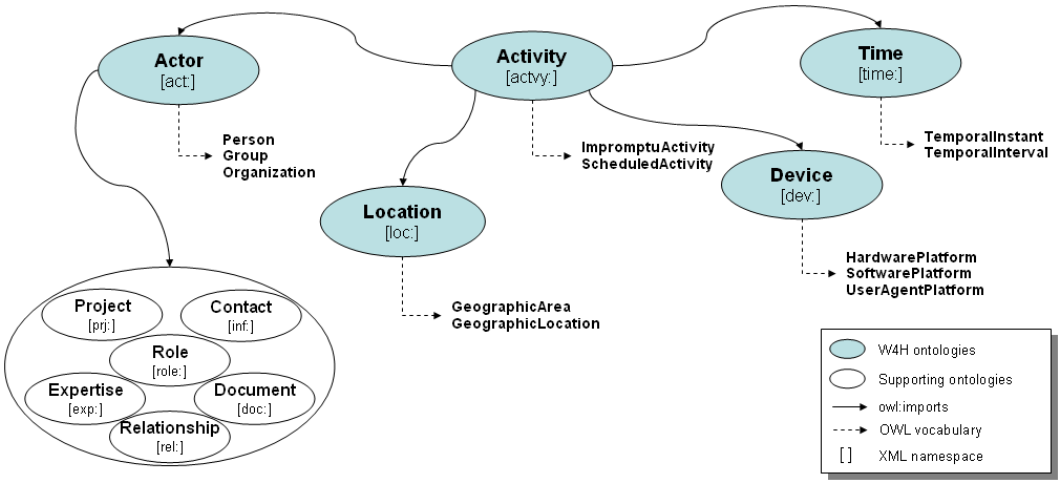

Figure 1. An overview of the domain-independent semantic context model with high-level ontologies.

also to geographic coordinates (e.g. latitude, longitude and altitude) combined with direction information (e.g. north, south and west). Absolute and relative location information is also modeled in this ontology.

Geographic locations are physical places where entities can be located such as room, floor and building. Geographic locations are classified into indoor and outdoor places. For instance, a bedroom is an indoor place, whereas a parking lot can be classified as an outdoor place.

Containment relations model atomic and composite places. Atomic places are those that can not be divided into separate parts, or its division is not relevant to a specific user-interaction. Composite places consist of separate interconnected parts, which can be atomic or even composite. Containment relations have importance, for instance, for detecting inconsistences regarding person's location: a person can not be in two different atomic places at the same time. If a location-aware system answers that a person is in two different places, either these must have a relation of composition between them (e.g. suite and bathroom), or there are imperfections sourced from sensing technology.

Spatial relations between geographic locations are also represented. A special type of spatial relation is the physical connectivity in which it is possible to assert whether two places are physically connected (e.g. connectivity between rooms and corridors). The *geographic area* concept represents geographic locations from the address's point of view. In other words, a place such as a building can be related to the address where it is located (e.g. street and zip code).

Upper ontologies such as OpenCyc [20] and SUMO (*Suggested Upper Merged Ontology*) [28] are the foundation for modeling the *Location* ontology. Example 2 shows a simple description of a place with corresponding location information.

```
<!--            Example 2            -->
<loc:Apartment rdf:ID="Ap22Std7Bldg">
  <rdf:type rdf:resource="&loc;IndoorLocation"/>
  <loc:locationName>Apartment 22</loc:locationName>
  <loc:isPartOf rdf:resource="#Std7Bldg"/>
  <loc:isConnectedTo rdf:resource="#Ap23Std7Bldg"/>
</loc:Apartment>
<loc:Building rdf:ID="Std7Bldg">
  <rdf:type rdf:resource="&loc;IndoorLocation"/>
  <loc:locationName>Studio 7</loc:locationName>
  <loc:isLocatedAt rdf:resource="#Std7BldgAddr"/>
</loc:Building>
<loc:GeographicArea rdf:ID="Std7BldgAddr">
  <loc:zipCode>13566-582</loc:zipCode>
</loc:GeographicArea>
```

Based on the *Location* ontology and the information in example 2, an inference engine can infer that:

- both individuals *Std7Bldg* and *App22Std7Bldg* are indoor locations as already asserted;
- the individual *Std7Bldg* represents a composite place because it has a component called *loc:Ap22Std7Bldg*. The *loc:isComposedOf* property is inferred according to its inverse property *loc:isPartOf*. Hence the individual *loc:Std7Bldg* is composed of the individual *loc:Ap22Std7Bldg*;
- the individual *loc:Ap22Std7Bldg* is located at the same address as *loc:Std7Bldg* because the former is a component of the latter;
- the individual *loc:Ap23Std7Bldg* is also connected to the individual *loc:Ap22Std7Bldg* because the property *loc:isConnectedTo* is modeled as *owl:SymmetricProperty*.

3.3 When-based ontology

Temporal information is everywhere in the real world. It can include dates (e.g. July 4, Friday), time duration of activities (e.g. attending a talk for 20 minutes), and ordering between events (e.g. beeping after a special incoming email). We already presented elsewhere [29] the development of our *Time* ontology, which was built in order to allow the reasoning about the temporal behavior of a user-interaction as well as to keep the history of its changes. The OWL-Time [30] and the SUMO [28] ontologies are the main foundation of our temporal ontology.

The *Time* ontology represents temporal information in terms of temporal instants and intervals. A temporal instant is a point on the universal timeline, whereas a temporal interval is delimited by two distinct and convex time instants. Temporal intervals are the core of our ontology. We have modeled some properties of intervals (e.g. the *durationOf*) and temporal relations between intervals as follows: *intEquals*, *intStarts*, *intFinishes*, *intMeets*, *intOverlaps*, *intFollows*, *intContains*, and their respective inverse relations *intStartedBy*, *intFinishedBy*, *intMetBy*, *intOverlappedBy*, *intPrecedes*, and *intDuring*. Observe that *intEquals* does not have an inverse relation.

We also model calendar and clock information to represent time in multiple granularities such as date, day of week, month, year, etc. XML Schema datatypes [25] provide a standard way of representing this type of information such as *xsd:gDay*, *xsd:gMonth* and *xsd:gYear*.

Example 3 is an excerpt of our temporal ontology. The class *TemporalEntity* is the superclass of temporal entities. The disjoint classes *TimeInstant* and *TimeInterval* are subclasses of *TemporalEntity*. The properties *beginPointOf* and *endPointOf* are modeled as functional properties because both must have only one value for each temporal instant. The property *insideOf* describes that a temporal instant is part of a temporal interval.

```
<!--            Example 3            -->
<owl:Class rdf:ID="TemporalEntity"/>
<owl:Class rdf:ID="TimeInstant">
  <rdfs:subClassOf rdf:resource="#TemporalEntity"/>
</owl:Class>
<owl:Class rdf:ID="TimeInterval">
  <rdfs:subClassOf rdf:resource="#TemporalEntity"/>
  <owl:disjointWith rdf:resource="#TimeInstant"/>
</owl:Class>
<owl:ObjectProperty rdf:ID="beginPointOf">
  <rdf:type rdf:resource="&owl;#FunctionalProperty"/>
  <rdfs:domain rdf:resource="#TemporalEntity"/>
  <rdfs:range rdf:resource="#TimeInstant"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="endPointOf">
  <rdf:type rdf:resource="&owl;#FunctionalProperty"/>
  <rdfs:domain rdf:resource="#TemporalEntity"/>
  <rdfs:range rdf:resource="#TimeInstant"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="insideOf">
  <rdfs:domain rdf:resource="#TimeInstant"/>
  <rdfs:range rdf:resource="#TimeInterval"/>
</owl:ObjectProperty>
```

3.4 How-based ontology

The *Device* ontology describes computational devices by means of profiles, which include a set of descriptions that model devices features regarding three platforms: *hardware*, *software*, and *user agent*. The hardware platform describes a device regarding its *input*, *output*, and *network* features. The input and output platforms model the I/O capabilities of devices: the type of keyboard and pointing devices supported, whether the display is color-capable, etc. The network platform represents the characteristics of network connections of a device such as whether it supports wireless connection, and the network speed.

The software platform represents the application environment, operating system, and installed software. For instance, it allows to describe whether the user accepts downloadable software, the types of Java virtual machines supported, and the current operating system. The user agent platform describes the software browser running on a device: name, version, and whether it is frames-capable and JavaScript-enabled.

Example 4 describes a short device profile of a PDA: the hardware platform is composed of the screen size in terms of pixels (*138x84*) and the number of characters (*17x5*), and the number of bits per pixel supported (*16*); the software platform is composed of the name (*Windows Mobile*) and the version (*2003 2nd Edition*) of the operating system, and whether the system is Java enabled (*true*).

```
<!--            Example 4            -->
<dev:ComputationalDevice rdf:ID="ipaqDvc">
  <dev:hasProfile rdf:resource="#ipaqPrf"/>
</dev:ComputationalDevice>
<dev:Profile rdf:ID="ipaqPrf">
  <dev:hasHardwarePlatform rdf:resource="#ipaqHP"/>
  <dev:hasSoftwarePlatform rdf:resource="#ipaqSP"/>
</dev:Profile>
<dev:HardwarePlatform rdf:ID="ipaqHP">
  <dev:ScreenSize>138x84</dev:ScreenSize>
  <dev:ScreenSizeChar>17x5</dev:ScreenSizeChar>
  <dev:BitsPerPixel>16</dev:BitsPerPixel>
</dev:HardwarePlatform>
<dev:SoftwarePlatform rdf:ID="ipaqSP">
  <dev:hasOSName>Windows Mobile</dev:hasOSName>
  <dev:hasOSVersion>2003 2nd Edition</dev:hasOSVersion>
  <dev:isJavaEnabled
   rdf:datatype="&xsd;boolean">true</dev:isJavaEnabled>
</dev:SoftwarePlatform>
```

A device profile like this should be able to answer how information, services, and user interfaces can be adapted using it. We have borrowed most concepts from the W3C recommendation for describing devices capabilities called CC/PP (*Composite Capabilities/Preferences Profile*) [21].

3.5 What-based ontology

The *Activity* ontology describes actions that people do or cause to happen. Due to the difficulty of perceiving and interpreting human actions, we have modeled an activity in

terms of the relevant set of events that characterizes it. In our approach, an event is a fact that includes *spatiotemporal* descriptions, as well as descriptions of the corresponding *actors* and *devices* involved. Events are temporally classified as instant events and interval events, i.e., those events related to time instants and time intervals, respectively. The relevance, the type and the combination of events for inferring activities are dependent on the user's task in the current domain.

Activities are modeled as of two disjoint types: *impromptu* and *scheduled*. The former represents activities that occurs in an informal manner (e.g. cocktail meetings), whereas the latter represents activities planned in terms of time and place (e.g. invited talks in a conference). Thus, the *Activity* ontology imports other ontologies of our context model (in this case, *actor*, *location* and *time*), as shown in **Figure** 1.

The RDF serialization in Example 5 describes a lecture activity taking place at the classroom 3010. Due to space restrictions, we describe only one event as part of that activity. In this case, we describe a person called "John Doe" attending a scheduled lecture for two hours (*PT2H* represented in XML Schema). It is also shown that "John Doe" is a student, he works with someone identified by "IanBattle", and he is using an electronic, color-capable whiteboard (SMBoardDvc). It is also specified that the classroom 3010 is part of a composite place identified by "Section2ICMC".

```
<!--            Example 5            -->
<actvy:Meeting rdf:ID="lecture3010">
  <rdf:type rdf:resource="&actvy;#ScheduledActivity"/>
  <actvy:IntervalEvent rdf:resource="#JohnAt3010"/>
</actvy:Meeting>
<actvy:IntervalEvent rdf:ID="JohnAt3010">
  <actvy:participant rdf:resource="#JohnDoe"/>
  <actvy:location rdf:resource="#Room3010Section2"/>
  <time:durationOf
   rdf:datatype="&xsd;duration">PT2H</time:duration>
</actvy:IntervalEvent>
<act:Person rdf:ID="JohnDoe">
  <act:hasName>John Doe</act:hasName>
  <act:hasRole rdf:resource="&role;Student"/>
  <act:ownsDevice rdf:resource="#SMBoardDvc"/>
  <rel:worksWith rdf:resource="#IanBattle"/>
</act:Person>
<loc:Classroom rdf:ID="Room3010Section2">
  <loc:locationName>Classroom 3010</loc:locationName>
  <loc:isPartOf rdf:resource="#Section2ICMC"/>
</loc:Classroom>
<dev:ComputationalDevice rdf:ID="SMBoardDvc">
  <dev:hasProfile rdf:resource="#SMBoardPrf"/>
</dev:ComputationalDevice>
<dev:Profile rdf:ID="SMBoardPrf">
  <dev:hasHardwarePlatform rdf:resource="#SMBoardHP"/>
</dev:Profile>
<dev:HardwarePlatform rdf:ID="SMBoardHP">
  <dev:bitsPerPixel>32</dev:bitsPerPixel>
</dev:HardwarePlatform>
```

The *Activity* ontology has reused concepts from RCal (*Retsina Semantic Web Calendar Agent*), which is an RDF vocabulary for schedules and events on the Web [31].

3.6 A lower ontology for e-learning

To illustrate the use of our model, we present a lower ontology for the e-learning domain on campus. The hierarchy of concepts depicted in **Figure** 2 represents the subclass relations between concepts of each ontology, not the properties and relations. Classes of the *Device*, *Time* and *Contact* ontologies are not shown because their concepts do not change according to the e-learning field. The extensions to our context model are depicted by vertical bars.

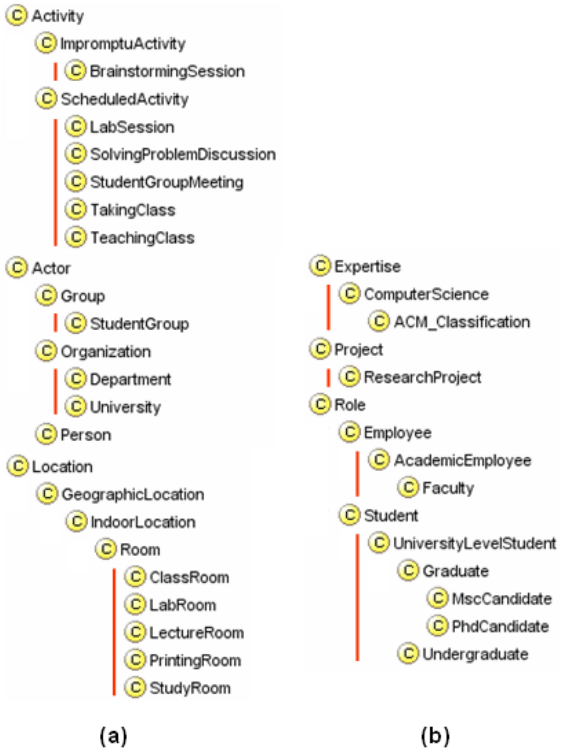

Figure 2. Vertical bars depict extensions of (a) the Activity, the Actor and the Location ontologies, and (b) the ontologies supporting the Actor ontology.

The *Relationship* and the *Document* ontologies are extended with respect to their properties only. For instance, the former includes the property *isSupervisorOf* between people whose roles are *Faculty* and *Graduate*, whereas the latter models the property *projectHomepage* between a *Project* and its corresponding *Homepage*, which in turn is subclass of *Document*.

As described in Section 3, our context model classifies activities as of two disjoint types: *ImpromptuActivity* and *ScheduledActivity*. Considering the e-learning domain, a common impromptu activity is *BrainstormingSession*. On the other hand, scheduled activities in the e-learning field

include: activities performed in laboratory as subclasses of *LabSession*, activities performed in classroom as subclasses of *TakingClass* and *TeachingClass* (depending on the role of the actor involved in), meetings between teaching assistants and students as subclasses of *SolvingProblemDiscussion*, and meetings between groups of students as subclasses of *StudyGroupMeeting*.

In the same context, actors are represented as *StudyGroup* (subclass of *Group*) and *Department* and *University* (subclasses of *Organization*). *Person* is also an actor with the following roles: *UniversityLevelStudent*, which in turn is classified as *Graduate* and *Undergraduate* students, and *AcademicEmployee*, which in turn can be a *Faculty* member. Projects in the campus environment can be extended as *ResearchProject*.

In order to represent people's expertise in a computer science course, for instance, we have built an OWL-based version of the ACM Classification System. There are two means of relating knowledge using this ontology: the hierarchical classes of knowledge, and the symmetric property *isKnowledgeRelatedTo* inherited from the *Expertise* ontology. Regarding the former approach, it is possible to infer that the class *knowledge representation formalisms and methods* is subclass of the class *artificial intelligence*. Considering the latter approach, it is possible to infer that the classes representing knowledge of *mathematical logic* and *knowledge representation formalisms and methods* are interrelated by means of the property *isKnowledgeRelatedTo*.

Finally, activities performed by actors on campus are likely to take place in the following subclasses of *IndoorLocation*: *ClassRoom*, *LabRoom*, *LectureRoom*, *PrintingRoom*, and *StudyRoom*. Those places are of the type atomic, i.e., they are not composed of other places.

The next section presents the semantic service infrastructure that we have built for the management of context.

4. The Semantic Context Kernel

The literature has reported the need for software infrastructures that make it easier the development of context-aware applications [2] [6]. We have implemented a service infrastructure based on our semantic context model called *Semantic Context Kernel*. The main objective is to provide application developers with a set of semantic-enabled services that can be configured so as to address applications' requirements. Since this work handles the semantics of context, it furthers our previous work on software infrastructure for context awarenesss [32]. **Figure** 3 depicts the Semantic Context Kernel architecture.

In **Figure** 3, context information is provided by *Context Sources*, which include applications, web services, and physical sensors. *Context Transducers* convert the information captured from context sources into a common semantic

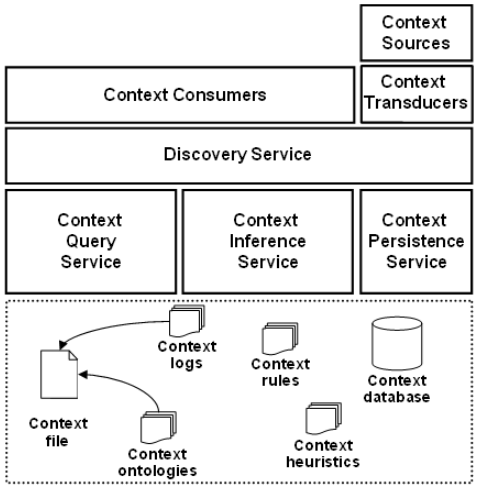

Figure 3. An overview of the Semantic Context Kernel architecture.

representation: the RDF triple model. This approach addresses both the interoperability and the reuse issues. *Context Consumers* make use of context information so that they can adapt themselves following the current situation (e.g. applications).

The *Discovery Service* provides context transducers and every service following described with an advertising mechanism so as to allow context consumers to locate these services. The *Context Inference Service* provides context consumers with a configurable inference support about context (e.g. transitive and rules-based inference). Context rules are defined by users and stored on files. Should the inference over context originates a conflict, user-defined *context heuristics* can be used to resolve it. The *Context Query Service* allows context consumers to query context through a declarative language for RDF models called RDQL (*RDF Data Query Language*) [33].

The *Context Persistence Service* allows the persistent storage of context in a configurable manner so that application developers can choose the types of persistent storage and context representation. We allows to store context both on relational databases and a context file. The types of RDF serialization supported include RDF/XML and N-Triples [34], which is a line-based, plain text format for encoding RDF graphs.

In the file-based approach, every information collected from a context source is firstly stored in memory. In order to prevent loss of information, it is then copied to context log files on a regular basis, and afterwards these log files are merged into the context file. Despite the amount of memory necessary to store large RDF models, the advantage of this approach is twofold: simplicity and good performance on query processing. In this approach, both the triples repre-

sentation of context and the content of ontologies are stored on the context file. Otherwise, when storing context on databases, ontologies are stored on separate files and only read when necessary (e.g. by the context inference service). User-defined context rules are stored on files and used along with RDF models and ontologies for the reasoning process.

5. Related Work

Making applications context-aware is one of the challenges pointed out in ubicomp. A key issue for achieving this is a high-level context model describing relevant characteristics of ubicomp interactions. This section presents the evolution of context models in the past few years, which differ in the classification of context, the scope of each kind of information, and the underlying modeling technique.

Pioneering efforts on context modeling represent context for specific applications [8] [9]. They are also encoded in a proprietary format which make it difficult context reuse and sharing. These models mainly exploit location information acquired by physical sensors. Conversely, our context model has been modeled to be instanced in several applications domains. It has also reused Semantic Web standards which encourage context reuse and sharing. In addition, the model we propose represents not only location information, but also actors, time, devices, and activities.

The HP Cooltown context model describes people, places and things in the form of HTML pages to show the current state of each entity [10]. As CoolTown is primarily aimed at supporting applications that display context and services to end-users, its context model lacks formality and expressiveness for context reasoning. On the other hand, our ontological model has been designed to support different types of context reasoning. In addition, our model represents temporal information in order to generate historical context, whereas Cooltown's model does not consider time.

A step further on context modeling includes XML-based context descriptions. The Context Toolkit handles XML representations of context gathered from physical sensors [3]. The Context Fabric infrastructure represents context and events by means of the XML-based *Context Specification Language* [11]. The *MediaObject* model employs XML Schema for describing context associated to multimedia content [12]. Despite those models cover a variety of information and take advantage of the XML interoperability aspect, the XML syntax lacks formality and expressiveness for context modeling and reasoning. In addition, the ability of context interpretation in those works is hard-coded. Conversely, our context model is a step further regarding context representation since we use standards for representing the semantics of context. Besides, the use of ontologies allows loose coupling between systems and the context information they manipulate.

More computer-understandable representations of context are achieved by means of conceptual data modeling approaches. A sensor fusion system represents context as Entity-Relationship models so as to facilitate the storage, the association, the update, and the retrieval of context [13]. Its model classifies context along three dimensions: space/physical, time/history, and spiritual/social relationship. Henricksen et al. [14] propose a context model based on a combination of ER and UML models. It represents five context dimensions, in a similar manner to our *W4H* approach: person, location, time, activity, and device. Both approaches also provide a large variety of context including temporal information. Despite this, their context models do not allow inference support over context.

Artificial intelligence techniques have been also used for context modeling. Ranganathan et al. [15] model context as first order predicates (FOP). The structure of different context predicates is specified in an ontology, and each context type corresponds to a class. In this case, the ontology is used to check the validity of context predicates. The FOP approach has some positive points: the model allows complex rules involving contexts, and it enables automated inductive and deductive reasoning about context. Besides, according to the authors, the set of operators and quantifiers of the FOP model provides the necessary mechanism for context modeling. In comparison to that work, our approach also models the same types of context and provides inference over context. On the other hand, our approach exploits the standard RDF triple model as context interchange format between context sources and context consumers.

The following are two works that exploit the RDF triple model and OWL ontologies for context modeling. The COBRA ontology is an attempt to provide a set of standard ontologies for ubicomp applications [7]. In comparison to our work, we also model context using OWL ontologies, and the context dimensions we use as design space are similar. The level of modularity between the COBRA ontologies differs with respect to our work: this results in a higher-level of context reuse. Another positive aspect of COBRA is the definition of both an ontology language and the implementation of an engine so as to control context sharing between applications. However, we propose supporting ontologies to assist system designers to model actors' profiles.

The CONON ontology is another effort toward providing a standard OWL-based ontology for ubicomp environments such as the meetings and home domains [5]. Furthermore, the CONON ontology addresses important issues such as context classification and dependency, and quality of context. Regarding the context dimensions modeled and the types of context for each dimension, we exploit temporal information and supporting ontologies for modeling actors' profiles. CONON models actors only as *User*.

6. Concluding Remarks

This paper proposed an ontological context model for context-aware computing that has been built by means of the following steps: (i) determining the *W4H* design space of context-aware applications — actors (*who*), location (*where*), time (*when*), activities (*what*), and devices (*how*); (ii) sketching a list of questions that the model should be able to answer for each dimension; (iii) reusing existing and consensus vocabularies; and (iv) developing ontologies as an iterative process. Observe that we did not import the vocabularies of existing ontologies due to the amount of information they describe. Hence we borrowed their concepts in order to avoid overload on the process of inference over context. Regarding the iterative ontology development, we have been continuously working on the *W4H*-based ontologies toward a domain-independent semantic context model.

We have been developing the Semantic Context Kernel infrastructure, which attaches semantics to context information based on our ontological context model. That service infrastructure allows applications to become "semantic enabled" by providing them with configurable services to store, query, and reason about semantic context. The Semantic Context Kernel and the semantic context model have been validated by means of applications from the e-learning and the CSCW domains. Our previous work exploited non-semantic context in both domains [35].

Our context model will be also evaluated by means of such applications domains with respect to its generality and suitability for context modeling, which in turn are related to the ontologies and vocabularies we have reused. For instance, it has recently shown [23] that the CC/PP vocabulary is not suitable for devices descriptions with respect to relationships between systems' components (e.g. compositions), data typing, and profiles resolution.

Some key points about context have not been considered yet in our work such as inconsistence, update, and privacy of context. In the Semantic Context Kernel architecture, we assume context sources as accurate and updated context providers. We should support privacy of context [36] in situations where it is necessary, e.g., when context consumers need to track someone's whereabouts. Access control lists can be good candidates to allow privacy of context in a future version of our context model.

Regarding the OWL language, we have learnt that its semantics lacks constructs for modeling composition (e.g. the containment relation between geographic locations) and the definition of rules. There is a W3C proposal of a standard rule language for the Semantic Web where OWL axioms are extended with Horn-like rules called SWRL (*Semantic Web Rule Language*) [37]. We believe that the Semantic Context Kernel can take advantage of SWRL rules with respect to semantic interoperability.

Acknowledgments

Renato Bulcão Neto is PhD candidate granted by the FAPEMA funding agency (n.03/345). This work is also funded by the *Applied Mobile Technology Solutions in Learning Environments Project*, which is a partnership between Hewlett Packard and the University of São Paulo.

References

[1] M. Weiser, "The computer for the 21st century," *Scientific American*, vol. 265, no. 3, pp. 94–104, 1991.

[2] G. D. Abowd, E. D. Mynatt, and T. Rodden, "The human experience," *IEEE Pervasive Computing*, vol. 1, no. 1, pp. 48–57, 2002.

[3] A. K. Dey, "Understanding and using context," *Personal and Ubiquitous Computing Journal*, vol. 5, no. 1, pp. 4–7, 2001.

[4] M. Philipose, K. P. Fishkin, M. Perkowitz, D. J. Patterson, D. Fox, H. Kautz, and D. Hahnel, "Inferring activities from interactions with objects," *IEEE Pervasive Computing*, vol. 3, no. 4, pp. 50–57, 2004.

[5] T. Gu, H. K. Pung, and D. Q. Zhang, "Toward an OSGi-based infrastructure for context-aware applications," *IEEE Pervasive Computing*, vol. 3, no. 4, pp. 66–74, 2004.

[6] S. Helal, "Programming pervasive spaces," *IEEE Pervasive Computing*, vol. 4, no. 1, pp. 84–87, 2005.

[7] H. Chen, T. Finin, A. Joshi, F. Perich, D. Chakraborty, and L. Kagal, "Intelligent agents meet the Semantic Web in smart spaces," *IEEE Internet Computing*, vol. 8, no. 6, pp. 69–79, 2004.

[8] B. N. Schilit, N. Adams, R. Gold, M. Tso, and R. Want, "The PARCTab mobile computing system," in *Proceedings of the Workshop on Workstation Operating Systems*, pp. 34–39, 1993.

[9] R. Want, A. Hopper, V. Falcão, and J. Gibbons, "The Active Badge location system," *ACM Transactions on Information Systems*, vol. 10, no. 1, pp. 91–102, 1992.

[10] T. Kindberg, J. Barton, J. Morgan, G. Becker, D. Caswell, P. Debaty, G. Gopal, M. Frid, V. Krishnan, H. Morris, J. Schettino, B. Serra, and M. Spasojevic, "People, places, things: web presence for the real world," *Mobile Networks and Applications*, vol. 7, no. 5, pp. 365–376, 2002.

[11] J. I. Hong and J. A. Landay, "An infrastructure approach to context-aware computing," *Human-Computer Interaction Journal*, vol. 16, no. 2-3, 2001. 21 pages.

[12] R. Goularte, E. S. Moreira, and M. G. C. Pimentel, "Structuring interactive TV documents," in *Proceedings of the ACM Symposium on Document Engineering*, pp. 42–51, 2003.

[13] H. Wu, M. Siegel, and S. Ablay, "Sensor fusion for context understanding," in *Proceedings of the IEEE Instrumentation and Measurement Technology Conference*, 2002. 5 pages.

[14] K. Henricksen, J. Indulska, and A. Rakotonirainy, "Modeling context information in pervasive computing systems," in *Proceedings of the International Conference on Pervasive Computing*, pp. 169–180, 2002.

[15] A. Ranganathan and R. H. Campbell, "An infrastructure for context-awareness based on first order logic," *Personal and Ubiquitous Computing Journal*, vol. 7, no. 6, pp. 353–364, 2003.

[16] T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web," *Scientific American*, vol. 284, no. 5, pp. 35–43, 2001.

[17] K. N. Truong, G. D. Abowd, and J. A. Brotherton, "Who, what, when, where, how: Design issues of capture & access applications.," in *Proceedings of the International Conference on Ubiquitous Computing*, pp. 209–224, 2001.

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

[19] D. Brickley and L. Miller, "FOAF vocabulary specification," 2005. http://xmlns.com/foaf/0.1/.

[20] OpenCyc.org, "OpenCyc: The Project," 2005. http://www.opencyc.org/.

[21] G. Klyne, F. Reynolds, C. Woodrow, H. Ohto, J. Hjelm, M. H. Butler, and L. Tran, "Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 1.0," 2004. http://www.w3.org/TR/CCPP-struct-vocab/.

[22] G. Klyne and J. J. Carroll, "Resource Description Framework (RDF): Concepts and Abstract Syntax," 2004. http://www.w3.org/TR/rdf-concepts/.

[23] R. Eisinger, M. G. Manzato, and R. Goularte, "Devices descriptions for context-based content adaptation," in *Proceedings of the 3rd Latin American Web Congress*, IEEE Press, 2005. 9 pages.

[24] 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," 2004. http://www.w3.org/TR/owl-ref/.

[25] P. V. Biron and A. Malhotra, "XML Schema part 2: Datatypes," 2004. http://www.w3.org/TR/xmlschema-2/.

[26] Dublin Core Metadata Initiative, "DCMI metadata terms," 2005. http://dublincore.org/documents/dcmi-terms/.

[27] R. Ianella, "Representing vCard objects in RDF/XML," 2001. http://www.w3.org/TR/vcard-rdf.

[28] A. Pease and I.Niles, "IEEE Standard Upper Ontology: A progress report," *Knowledge Engineering Review*, vol. 17, pp. 65–70, 2002.

[29] R. F. Bulcão Neto and M. G. C. Pimentel, "Semantic interoperability between context-aware applications," in *Proceedings of the IX Brazilian Symposium on Multimedia and Web Systems*, pp. 371–385, 2003. (In Portuguese).

[30] J. R. Hobbs and F. Pan, "An ontology of time for the Semantic Web," *ACM Transactions on Asian Language Processing (TALIP)*, vol. 3, pp. 66–85, March 2004.

[31] T. R. Payne, R. Singh, and K. Sycara, "Calendar agents on the Semantic Web," *IEEE Intelligent Systems*, vol. 17, no. 3, pp. 84–86, 2002.

[32] C. R. E. Arruda Jr, R. F. Bulcão Neto, and M. G. C. Pimentel, "Open context-aware storage as a web service," in *Proceedings of the 1st International Workshop on Middleware for Pervasive and Ad-Hoc Computing, 4th ACM/IFIP/USENIX International Middleware Conference*, pp. 81–87, 2003.

[33] L. Miller, A. Seaborne, and A. Reggiori, "Three implementations of SquishQL: a simple RDF query language," in *Proceedings of the International Semantic Web Conference*, pp. 423–435, 2002.

[34] J. Grant and D. Beckett, "RDF Test Cases," 2004. http://www.w3.org/TR/rdf-testcases/#ntriples.

[35] R. F. Bulcão Neto, C. O. Jardim, J. A. Camacho-Guerrero, and M. G. C. Pimentel, "A web service approach for providing context information to CSCW applications," in *Proceedings of the 2nd Latin American Web Congress*, pp. 46–53, IEEE Press, 2004.

[36] S. Lahlou, M. Langheinrich, and C. Rocker, "Privacy and trust issues with invisible computers," *Communications of the ACM*, vol. 48, no. 3, pp. 59–60, 2005.

[37] I. Horrocks, "SWRL: a Semantic Web rule language combining OWL and RuleML," 2004. http://www.daml.org/rules/proposal/.