Expert Systems with Applications 41 (2014) 2859–2873

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

On the practicalities of place-based virtual communities: Ontology-based querying, application architecture, and performance

Tuan Nguyen [a], Seng W. Loke [b,*], Torab Torabi [b], Hongen Lu [b]

[a] University of Information Technology, Ho Chi Minh City, Vietnam
[b] Department of Computer Science and Computer Engineering, La Trobe University, VIC 3086, Australia

ARTICLE INFO

Keywords:
Place-based virtual communities
Place ontologies
Mobile services
Knowledge-based services

ABSTRACT

While the Internet has allowed geographical boundaries to be transcended, with the increasing use of the mobile Internet, there is a shift towards a focus on locality and place-specific applications. This paper proposes a novel approach for constructing context-aware mobile services for a place using a commonly shared knowledge base, that captures not only static but dynamic aspects of a place. The approach is based on a conceptual model of a Place-Based Virtual Community (PBVC), represented using an ontology; a PBVC for a place augments the place with context-aware services based on querying an ontology. We present an implementation of a framework based on the ontology and an evaluation of the performance of queries over the ontology. We also illustrate architectures of specific applications as specialisation of a generic PlaceComm architecture.

1. Introduction

Mobile devices such as PDAs, mobile phones and pagers have woven themselves into people's lives. There is a trend towards a virtual community co-existing with a physical place. In other words, places contain not only physical objects, real people, but also digital services and digital devices, forming, increasingly, over a place, a kind of Place-Based Virtual Community (PBVC). This paper identifies and addresses the need for capturing, storing, and understanding the context dependencies between a physical place and the social context of users within a community in order to build context-aware place-based services. In previous work (Nguyen, Loke, Torabi, & Lu, 2011), we proposed a systematic approach for developing place-based context-aware applications via the introduction of the PlaceComm framework, namely, we build an ontology, and then encapsulate the queries over the ontology as agents to provide services. A key distinguishing idea of this work is that the ontology also serves as a "standardized" schema for knowledge about a place. The purpose of such a "standardized" schema is that different context-aware applications for a place can be programmed against this schema, and different places can be expected to have the kind of knowledge mentioned in the schema.

This paper extends previous work as follows. We provide an updated description on the PBVC ontology and an updated description on the PlaceComm architecture, compared to what was presented in Nguyen et al. (2011). This paper presents more details on the format of items in the knowledge base, and presents in detail a comprehensive series of queries to illustrate the ideas of the work, not previously described elsewhere. We address questions and research gaps relating to the practical aspects of the PlaceComm approach to mobile services, namely, (i) what queries can be used for PBVCs and whether such queries can be efficiently evaluated against the ontology we developed, and (ii) what architectures for applications that use PlaceComm and how they relate to the generic PlaceComm architecture (e.g., whether the architectures of specific applications be viewed as a specialisation of the generic PlaceComm architecture). Also, here, we discuss trust aiding mechanisms and place discovery, not in previous work. This paper provides a comprehensive performance evaluation not previously published elsewhere. This paper relates the applications (in Section 5) to the generic PlaceComm framework in a new way, showing the specific architectures and how they are derived from the generic PlaceComm architecture. We demonstrate the system's versatility by developing prototype applications based on our implemented framework.

As advocated by Gordon and Silva (2011, 2012), this paper highlights the "return of focus" to locality and place that the mobile Internet is enabling.

The paper is organized as follows. Section 2 reviews the background for the paper. Section 3 details the notion of Place-based Virtual Community, from definition to theory and representation in the form of a PBVC ontology built using the Web Ontology Language (OWL).[1] Section 4 details the PlaceComm framework from its

* Corresponding author. Tel.: +61 394793415.
E-mail addresses: tuanna@uit.edu.vn (T. Nguyen), s.loke@latrobe.edu.au (S.W. Loke), t.torabi@latrobe.edu.au (T. Torabi), h.lu@latrobe.edu.au (H. Lu).

[1] The Web Ontology Language, http://www.w3.org/2004/OWL/.

0957-4174/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.eswa.2013.10.021

conceptual model to its agent architecture. Section 5 introduces a number of application prototypes built on the PBVC framework. Section 6 presents further evaluation of the PlaceComm framework, including performance measurement and scalability of querying. We conclude the paper in Section 7 with future directions.

2. Background and related work

2.1. Context-aware frameworks and services

There are detailed surveys on context-aware frameworks and services, e.g., in Baldauf, Dustdar, and Rosenberg (2007), Endres et al. (2005), Sheng, Yu, and Dustdar (2010) and Raz, Juhola, Serrat-Fernandez, and Galis (2006). Our work differs from earlier context-aware middleware and toolkits (Dey, 2001) in that these toolkits and middleware generally do not support reuse of a common knowledge base across different applications or allow applications to contribute knowledge. Site-specific services accessed via smartphones were considered in Toye, Sharp, Madhavapeddy, and Scott (2005), but they do not consider the notion of a community and a place KB.

2.2. Context modelling and reasoning using ontologies

There are many research projects that apply ontologies for context modeling and reasoning in context-aware computing (Ye, Coyle, Dobson, & Nixon, 2007; Bettini et al., 2010). Below, we review some significant projects that inspire and motivate our research.

2.2.1. SOUPA – standard ontology for ubiquitous and pervasive applications

In Chen, Perich, Finin, and Joshi (2004) is an ontology called Standard Ontology for Ubiquitous and Pervasive Applications (SOUPA). SOUPA provides a formal way to model context (Ye et al., 2007). SOUPA includes common vocabularies for BDI agents, time, space, events, user profiles, actions and policies for security and privacy.

2.2.2. CONON – the CONtext ONtology

CONON is an ontology-based context model in which a two-layer ontology approach is adopted for designing context ontologies (Wang, Zhang, Gu, & Pung, 2004). In Wang et al. (2004) is noted that even though pervasive computing environments such as the home, office or vehicle are different from each other, they still share common generic context models.

2.2.3. Smart-Context ontology

The project “Smart-Context” (Moore, Hu, & Wan, 2008) proposes a context ontology for pervasive computing which supports autonomous decision-making. The Smart-Context ontology is built using RDF/S with OWL and Jena, which support context reasoning.

2.2.4. PECO ontology

Recent work in Niu and Kay (2010) built a three layer ontology called PECO, which is a key component in the PERSONAF framework. The three layers of the PECO ontology are: (i) Middle Ontology, which represents key concepts about buildings, (ii) Application Ontology, which represents a particular building, and (iii) Accretion Ontology, which is a novel ontology that supports personalization. The contribution of the PECO ontology is its support for personalization in the sense of different people or groups of people being allowed to name a physical place with different names. For example, for postgraduate students “Room 125 is a Social Hub”, while for Bob “Room 125 is a Recharging Corner”.

2.2.5. Discussion

The literature shows that using ontologies is considered a key approach for modeling and reasoning about context. We propose our PBVC ontology later, which is built upon concepts in the above reviewed ontologies; however, we also introduce new concepts that relate to the notion of PBVC and show how the new concepts relate to concepts in other ontologies.

2.3. The importance of the notion of place

Recently, the notion of place has gained greater attention from the pervasive computing community. In Ciolfi (2004), Ciolfi and Bannon (2005) and Ciolfi, Fitzpatrick, and Bannon (2007), the authors have considered place as the environment for pervasive computing. In recent years, short-range wireless networks have pervaded urban places. The millions of wireless access points deployed in public places.[2] can lead to “Location-based Electronic Communities” as defined in Loke (2002). Indeed, many projects have investigated the appearance of devices, people and wireless activities in places, such as Cityware,[3] and Realtime City at MIT Senseable Lab.[4]

As noted above, there have been numerous work on context-aware frameworks (Endres, Butz, & MacWilliams, 2005), context ontologies (Chen et al., 2004; Blackstock, Lea, & Krasic, 2007) and context modeling and reasoning (Bettini et al., 2010). However, in our work, the notion of “place” is central, and knowledge about a place can be built, even without committing to any application initially. Other context ontologies can indeed complement our work.

3. The PBVC ontology

The PBVC ontology was first mentioned in Nguyen, Loke, Torabi, and Lu (2009). We published initial ideas on the concepts, architecture, and ontology of the PBVC in Nguyen, Loke, and Torabi (2007), Nguyen, Loke, Torabi, and Lu (2008), Nguyen, Loke, Torabi, and Lu (2010) and Nguyen, Loke, Torabi, and Lu (2010), but this paper provides a more comprehensive and update-to-date view of the ontology. Although, in building the PBVC ontology we incorporated concepts from other ontologies, the core part of the ontology is novel. Also, we identified particular ontologies that could be aggregated for our purposes and show how to employ the PBVC ontology in place-based services.

A place can be a stadium, a shopping mall, a street (e.g., Orchard Road in Singapore or the Bourke Street Mall in Melbourne, Australia), a neighborhood, the city center (e.g., Central Business District of a city), a museum, a park, a university, a collection of buildings, and so on. Each place can be associated with at least one PBVC which has information about the physical place and the community of people living in or being at that place. In general, the boundaries around a place may be specified by an n-sided polygon, where each point is represented by a set of n GPS coordinates, or a circle with given GPS coordinates for a point and a radius. Note that our notion of PBVC does not aim to replace but can complement existing virtual communities that transcend boundaries. More generally, a place can be a union of polygons (i.e., a set of disconnected areas). For example, a place labeled as “myCommonPlaces” is a union of the university campus, my home, my parent’s home and two shopping areas frequented.

Our vision is that each PBVC contains knowledge about a place, and the community at that place, a shared repository which we call

[2] http://www.skyhookwireless.com/howitworks/coverage.php.
[3] http://www.cityware.org.uk.
[4] http://senseable.mit.edu/.

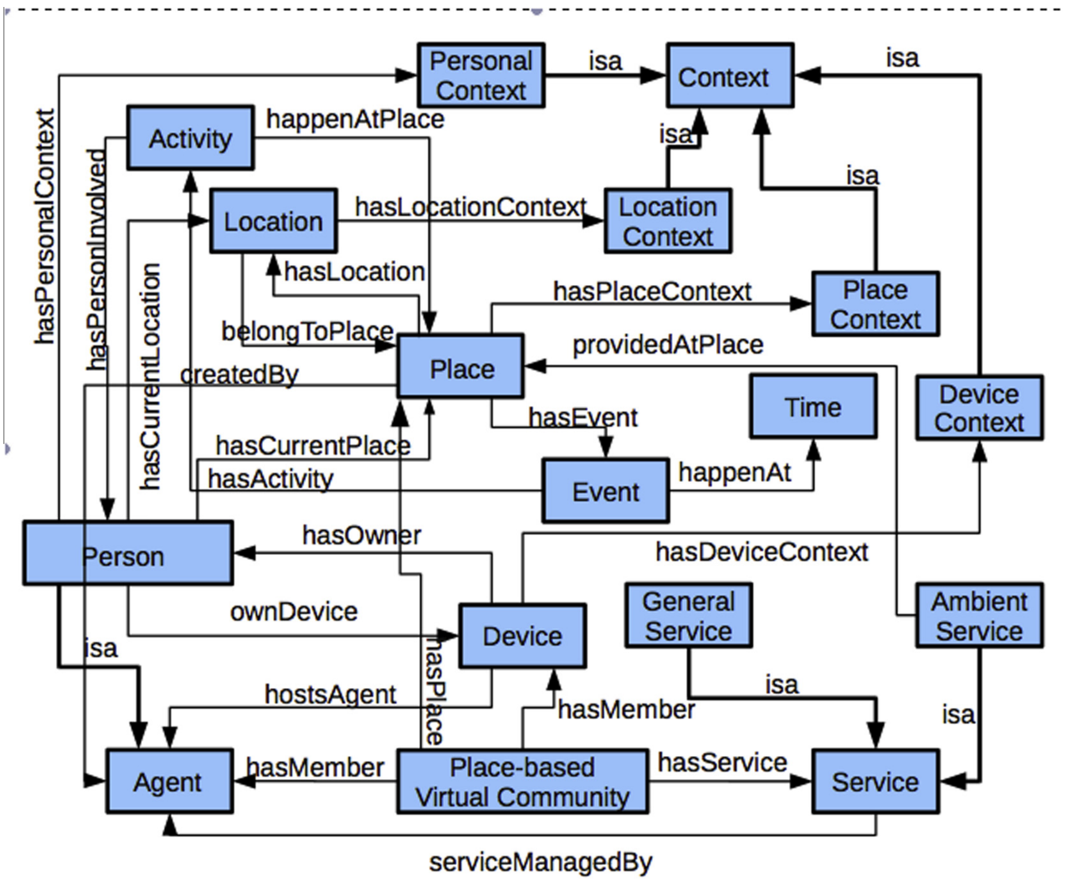


Fig. 1. Overview of the PBVC ontology.

a *place knowledge base* (or place KB, for short). Each person, via his or her device (s) might use such knowledge (or such knowledge used by a context-aware application on the device) or might contribute such knowledge back to the place KB. Different applications running on users' mobile devices can consume and/or contribute knowledge to the place KB. A place KB may contain knowledge of a more static nature, such as locations of buildings, or knowledge of a more dynamic nature, such as the current visitors at the place. It can also contain knowledge about physical features of a community, including restaurants and shops, available physical and digital services (e.g., electronic directory or virtual tour guide for a place), where people are assembling at different points in time within the place, as well as, with permission, even locations of particular people, a representation of relationships between people, and people and objects (e.g., ownership). Even a history of how people move through the place can be stored. Since the effective area of a PBVC is bounded by the place, there could be multiple overlapping PBVCs, each with their own KBs (e.g., the PBVC of a cafe is located within the PBVC of a street on which the cafe is located (Nguyen et al., 2007; Nguyen, 2008)). Consider a range of examples of context-aware mobile services applicable to a place, such as a virtual guide for the blind, a virtual tourist guide, a shopping-recommender system, a crowd-seeker, an ATM Bank Teller finder (with real-time reports of lengths of queues in front of them), and a navigation application. These applications may share knowledge about a place.

3.1. Modular design of the PBVC ontology

In developing our ontology for place knowledge, we consider the minimal range of concepts to describe a place and its dynamics (the ontology could be extended in future work). The PBVC ontology includes eleven main concepts: Agent, People, Device, Context, Place, Location, Service, Activity, Event, Time and Place-Based Virtual Community. The PBVC ontology is summarized in Fig. 1.

3.2. Inherited ontologies

The PBVC approach enables a common knowledge base for a place that can be shared and extended by users' contributions in order to maximize context re-use, avoiding the re-engineering of context modules in different applications. To build this knowledge base, we leveraged on other ontologies mentioned earlier, and some concepts which we have borrowed from other ontologies have been refined.

3.2.1. Person & FOAF ontology

The Person and Agent concepts in the PBVC ontology are inherited from the FOAF ontology.[5] The FOAF project defines concepts for capturing personal information and their relationships in social networking.

3.2.2. Service ontology

The service ontology is adapted from the OWL-S ontology (Martin et al., 2007). We have created two subclasses of Service which are GeneralService and AmbientService (Fig. 2). The GeneralService is a kind of service provided in the PBVC regardless of the place. The AmbientService is a kind of service that is provided at a specific place and may be different in each place. The services belonging to GeneralService exist in any PBVC community, while

[5] http://xmlns.com/foaf/spec/.

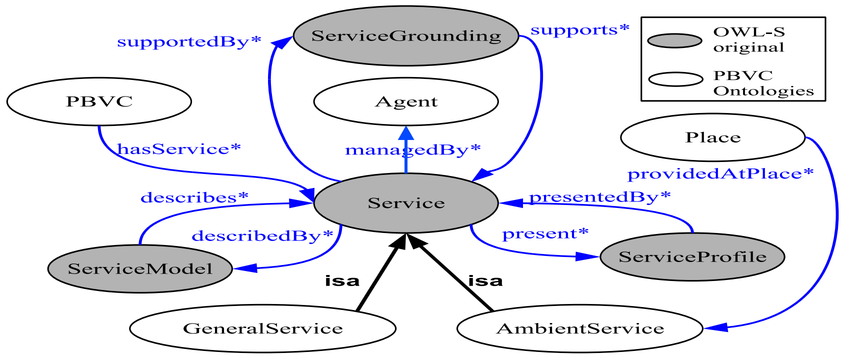


Fig. 2. Service ontology.

an AmbientService may be different at different places and communities. Each service has properties that allow service matching.

3.2.3. Device ontology

FIPA and W3C have defined different device ontologies for different purposes. While FIPA's device ontology looks at a device as the platform that can host an agent,[6] W3C's goal is to describe a device suitable for displaying a particular Web page.[7] However, in our work, a device can be any object such as a computer, a smartphone, a kind of furniture or even an RFID tag, i.e., the device ontology should facilitate describing a wider range of objects. Furthermore, a device can have an owner (hasOwner) or a person can own the device (ownDevice). By modelling this, we can discover (though with some uncertainty) a person's presence by his/her owned devices' presence.

3.2.4. Time ontology

Time is an important context information for all events happening in the environment. We inherited the OWL-Time ontology defined by Hobbs and Pan (2004) for our ontology. Also, by timestamping facts in the KB, we can allow querying about past events. For example, the question "Where is Tom?", has different answers at different times.

3.2.5. Location ontology

Location has been considered important in many applications (Hightower, Consolvo, Lamarca, Smith, & Hughes, 2005). We used the SOUPA location ontology for our system.

3.3. PBVC ontology

3.3.1. Place ontology

Place is a central concept in the PBVC ontology. By providing semantic links from place to other concepts, such as people, device, and events (i.e., with properties such as hasPersonPresence, hasDevice, hasEvent, hasObject, and belongTo), we can get a more meaningful description for a place. For example, this is 'my place', this is 'La Trobe postgraduate students' place' or even this is a 'place to relax'.

3.3.2. Activity ontology

We focus on social activities involving multiple users. Based on the PBVC ontology, we can build a knowledge base about activities at a place. For example, by analyzing the sensed data using the knowledge base, we can discover when a group activity is occurring. A scenario is illustrated in Fig. 3, where people's presence is detected by sensors, allowing us to examine Tom's social relationships with the people at that place, namely Tom's parents, his son and his wife. In addition, with the TV and DVD turned on, we can infer that they could be watching TV together. In addition, if we trace the past, where Tom has been, another meaningful context might be inferred: a reunion party.

3.3.3. Context ontology

The context ontology here is used as a repository for storing history. Listing 1 shows a snapshot of context of Room 219 in building PS1 at timestamp 1252524031. The timestamp follows the unix timestamp, which is milliseconds since 0 h 0 m 0 s on 1/1/1970.[8]

The place named "PostGradOfficePS1R219" has *PlaceContext* "ctx_PostGrad-OfficePS1R219". This instance will capture all the context of the room 219. For example, in this snapshot (Listing 1), the room 219 has two devices present: ChuongVoPhone, and TuanNokiaN95. All have the same timestamp value of 1252524031. Because the Device ontology and Person ontology are connected via the relationship *ownDevice*(Person,Device) and *belongTo (Device, Person)*, we can reason that there are two people in the room. This context is captured in the context KB, which means that it can be reused for later queries as well as for different applications.

Listing 1. Place Context in OWL.

```
<PlaceContext rdf:ID="ctx_PostGradOfficePS1R219">
   <hasTimeInstance>
     <TimeInstance rdf:ID="TimeInstance_11">
       <timestamp rdf:datatype="http://www.w3.org/2001/XMLSchema#
           string" >
       1252524031</timestamp>
     </TimeInstance>
   </hasTimeInstance>
   <plcCtx_hasDevicePresence rdf:datatype="http://www.w3.org
       /2001/XMLSchema#string">
   ChuongVoPhone</plcCtx_hasDevicePresence>
   <plcCtx_hasDevicePresence rdf:datatype="http://www.w3.org
       /2001/XMLSchema#string">
   TuanNokiaN95</plcCtx_hasDevicePresence>
   <plcCtx_PlaceID rdf:datatype="http://www.w3.org/2001/XMLSchema
       #string">
   PostGradOfficePS1R219</plcCtx_PlaceID>
   <timestamp rdf:datatype="http://www.w3.org/2001/XMLSchema#
       string">
   1252524031</timestamp>
 </PlaceContext>
```

We note that the timestamps provides an efficient solution to deal with consistency issues, in that, two contradictory (if viewed

[6] http://www.fipa.org/specs/fipa00091/PC00091A.html.
[7] http://www.w3.org/Mobile/CCPP/.
[8] http://www.unixtimestamp.com/index.php.

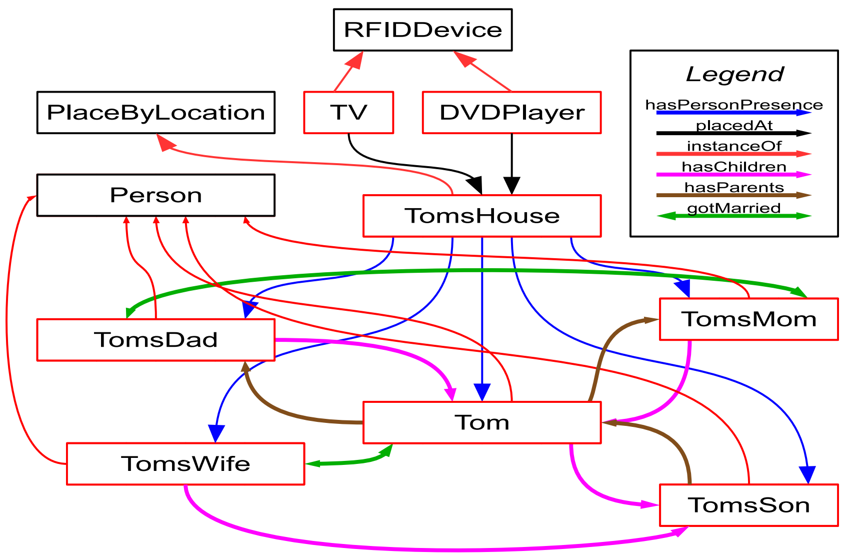


Fig. 3. Reunion party at Tom's house.

without timestamps) facts are no longer contradictory if viewed with their different timestamps.

3.3.4. Event ontology

In our work, an event can have many activities, which may or may not include people. In other words, some events can happen without the intervention of people. In addition, an event can be organized and involve many people. For example, the cultural event "Hello-Vietnam" is organized by Vietnamese students at La Trobe University (Listing 2). By combining people, place, and time factors, we can infer about the social activities that happened at the Agora.[9]

Listing 2. Hello Vietnam event.

```
<Event rdf:ID="HelloVietnamEvent">
 <date rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime"
 >2009-07-19T09:45:10</date>
 <hasPersonInvolved rdf:resource="#TuanNguyen"/>
 <hasPersonInvolved rdf:resource="#ChuongVo"/>
 <happenAtPlace rdf:resource="#Agora"/>
</Event>
```

3.3.5. Overall view: PBVC ontology

The PBVC ontology is illustrated in Fig. 1, which makes explicit the semantic links between concepts that allow us to extract relevant context information. We show that contexts modeled via these concepts of our PBVC ontology are useful in applications in Section 5.

3.3.6. Support for reasoning

The PBVC ontology supports context reasoning by exploiting the semantic links between classes. For example, a person registers his/her Bluetooth ID with the community. via the presence of a bluetooth device, we can infer about a person present by the relationship belongTo (device, Person). Listing 3 shows an OWL snapshot which describes the relationship between device "TuanNokiaN95" and person "TuanNguyen".

[9] Agora is the central place of La Trobe University, Bundoora campus.

Listing 3. Device details.

```
<Device rdf:ID="TuanNokiaN95">
   <deviceID rdf:datatype="&xsd;string">001ADCD98E21</deviceID>
   <deviceName rdf:datatype="&xsd;string">TuanNokiaN95</deviceName
       >
   <hasOwner rdf:resource="#TuanNguyen"/>
</Device>
```

Listing 4 shows the PostGradOffice place with no devices present.

Listing 4. An empty place named "PostgradOffice".

```
<Place rdf:ID="PostgradOffice">
   <placeName rdf:datatype="&xsd;string">PostgradOffice</placeName
       >
</Place>
```

After a device has appeared and is detected by the sensor agent in the room, the place is then updated with new information (Listing 5).

Listing 5. A device is detected at the PostgradOffice.

```
<Place rdf:ID="PostgradOffice">
       <hasDevicePresence rdf:resource="#TuanNokiaN95"/>
       <placeName rdf:datatype="&xsd;string">PostgradOffice</
           placeName>
</Place>
```

As we note later, a system will need to maintain knowledge about the PostgradOffice, and for example, check the knowledge base to discover the owner of the device "TuanNokiaN95" via the semantic link "hasOwner" between device and person. After getting the result, the place agent updates the knowledge base with new information (Listing 6). In this scenario, we do not detect a person carrying a wrong device or the device has been stolen.

Listing 6. Place have device and person presence.

```
<Place rdf:ID="PostgradOffice">
       <hasDevicePresence rdf:resource="#TuanNokiaN95"/>
       <hasPersonPresence rdf:resource="#TuanNguyen"/>
       <placeName rdf:datatype="&xsd;string">PostgradOffice</
           placeName>
</Place>
```

3.3.7. PBVC ontology support for place and social understanding

Related work on place discovery has shown that there are many different ways to discover a place such as in PlaceLab (Calabrese, Kloeckl, & Ratti, 2009; Hightower et al., 2005; Niu & Kay, 2008). In this section, we review and introduce five ways to recognise or identify a place via extracting knowledge from the KB using SPARQL queries, i.e., a place can be determined by: (i) its meaningful reverse geocoding address, (ii) a tag which can refer to any significant building or object, (iii) a particular person or people of a community, (iv) an event, and (v) its rhythm (as we explain below).

3.3.8. Place by location

A popular way to describe a place is by finding its meaningful postal address. In this framework, we use the Google Maps Reverse Geocoding Web service,[10] because it is freely available and easy to use. The user's location is sensed and contributed to the KB via the process of using the Placecomm framework.

3.3.9. Place by tagging

The second approach of detecting a place is by finding a significant object that can represent a place. A tag for an object can be anything that is electronically detectable such as a Bluetooth device, a wireless access point MAC address, RFID tags or even via GSM signals fingerprinting. Using PlaceLab "stumbling services" (Hightower, LaMarca, & Smith, 2006), we can detect a place and store it into the knowledge base. When there is a query that needs to resolve a place, we send a query to the KB to find out the place name. The idea of this approach can be expressed informally: "if I am near this object, tell me where I am."

3.3.10. Place by people

Place can be defined by people or a group of people or a community (Cresswell, 2004). This kind of place usually does not use location as a primary parameter. For example, "I am invited to my friend's house party", or "I am at Tim's place".

3.3.11. Place by event

At different times of the day or year, the place has different numbers of people attending. The Melbourne Cup event, for example, is usually hosted at the Flemington Racecourse in Melbourne once every year in the first week of November. We call that place the Melbourne Cup Event, thereby: identifying the place by an event. Interestingly, by examining the results from a KB of Bluetooth discovered devices, which we built over three years (2007, 2008, and 2009), it is found that there are devices re-appearing at the Flemington Racecourse in 2007 and 2008. The place Flemington Racecourse, hence, can be associated with an event - a fairly strong attachment of event and event attendees to a place.

3.3.12. Place by its rhythm

If we look at a road segment as a place, traffic at the road segment is not the same every time. Depending on the time of the day/year, traffic will be different. Based on the average speed of vehicles traveling on that road, we can discover the traffic situation on that road. We call the road segment "the jammed road" thereby: identifying the place by what we call its rhythm.

[10] http://code.google.com/apis/maps/documentation/geocoding/.

3.3.13. Place "without location"

Some places do not require a geographical location to be identified, for example, when using digital tagging location, especially when discovering place by tag. Just imagine when a mother is looking for her children, she receives an answer that her children are close to their grandparents. The mother then feels relieved because her children are in safe hands in a safe place, without needing to know the actual geographical location of the safe place.

3.4. Evaluation of the PBVC ontology

In Gómez-Pérez, Juristo, and Pazos (1995) is proposed a suitable method for evaluating ontologies, which contains: requirements specifications, competency questions, and applications building using our ontology. Firstly, in the requirements specifications step, we review the ontologies for pervasive computing, which have been described in Ye et al. (2007) and Bettini et al. (2010). We have selected vocabularies that are adequate to describe the context-aware domain, based on the place as the environment. The social aspect of users and places are also mentioned in the ontology. These features distinguish our contributions from other related work. The PBVC ontology is represented in OWL. Secondly, we built twenty-four competency questions which can be answered by using the PBVC ontologies (not all shown in this paper). Most of the competency questions can be answered directly by SPARQL queries. Thirdly, the PBVC ontology serves as a common vocabulary for the place domain, and also enables a knowledge base that can be shared by different applications based at the same place (as discussed further in Section 5).

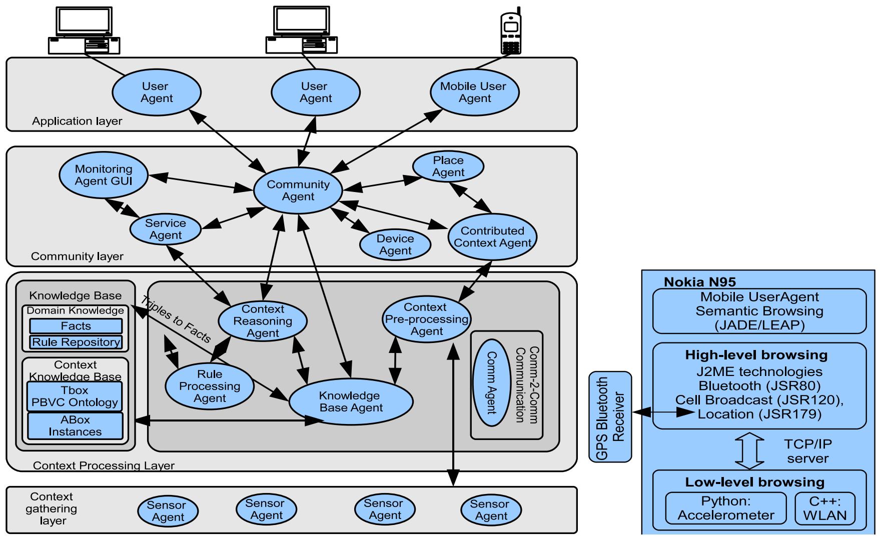
4. Implementation of the PlaceComm framework

This section details the PlaceComm (multiagent) framework, which is built based on the PBVC concepts. The ontology and framework were first discussed in our previous paper (Nguyen et al., 2011), but here, we provide a more update-to-date description. PlaceComm has been prototyped using a multiagent architecture with the JADE platform (Bellifemine, Poggi, & Rimassa, 2001). In previous work, we implemented the protocols for members joining and leaving a PBVC (Nguyen et al., 2007).

The layered architecture of the framework is given in Fig. 4(a). Applications in the application layer make use of services enabled by agents in the agent community layer. Services in the community layer include services for leaving and joining the community, communicating with other people in the community, managing connections with mobile clients, and monitoring services for obtaining information about the community.

The first/lowest layer is the **context-gathering layer**. It contains physical sensors managed by agents that can sense the environment to get context information.

The second layer is the **context-processing layer**. This layer takes care of all activities related to context. The Context Preprocessing Agent receives sensed context information from both Sensor Agent and User Agent as well as preprocesses them before passing the information to the Knowledge Base Agent (KBA) to insert into the Context Knowledge Base (CKB). The core component here is the CKB which is built on top of the PBVC ontology. The place KB effectively stores what we call context information (information about entities within the community, e.g., locations, etc. which are time-stamped) and also the more static information about the community (which might be entered at development time). In our prototype, we use the Sesame API and Google Maps



(a) PlaceComm Architecture (b) MUA

Fig. 4. The PlaceComm architecture in detail.

(if required) in managing information, and accessing maps. The Context-Reasoning Agent uses the Jess rules engine and the Knowledge Base Agent to answer complex queries as well as provide reasoning services for applications. The CommAgent, which is shown in Fig. 4(a), is in charge of community-to-community communications realizing the community stack concept (Nguyen et al., 2007). The Domain Knowledge component represents the rule-based understanding of sensed data. It is expressed using the Jess rule engine (Hill, 2003). The Domain Knowledge (DK) part represents specific knowledge about the domain and can be added later by users or by experts.

When a new context is inserted into a knowledge base, it also triggers the Context Pre-processing Agent to deduce more context (using Domain Knowledge) at a higher level and this context is also inserted into the KB according to predefined rules. Conversely, when a low level primitive context is removed (e.g., when a device is no longer detected), a higher level context fact is then also removed (e.g., the person is no longer present). Update to context occurs in response to events happening in the real world as detected by sensors, and so, is event-driven.

We define *cascading insertion* as: when a new context is inserted into a knowledge base, it also triggers the context pre-processing agent to deduce more context at a higher level and this context is also inserted into the KB according to predened rules.

Cascading insertion can trigger a chain reaction and lead to unending insertions. Therefore, we propose a particular procedure for cascading insertion:

1. Check if the fact is not already inserted, and if not, insert context into the KB; then trigger the cascading insertion for that context.
2. Deduce further context from Step 1 according to a predened rule, then insert the newly deduced context into the KB.
3. Check conditions for stopping (e.g., if there are still rules that apply), otherwise, go to Step 2.

Consider an example. The sensor agent in the SEMS Room detected two phones “TomNokiaN95” and “PeterPhone”. Based on the relationship *hasOwner*(Device,Person) in the PBVC ontology, another process is created and the new context information “Tom and Peter are in the room” is inserted. By examining their social relationships *friendOf* and *belongTo*, the system knows that they are friends of each other and all are members of *PerComGroup*. In addition, other objects in the room such as Projector1, Smartboard and Computer are turned on, and combined with the function of the SEMSRoom for doing seminars, the system may continue to insert new facts: that there is a PerComGroupSeminar in the SEMSRoom.

Opposite to cascading insertion, the process of *cascading deletion* is defined as a process of improving the KB quality to delete less meaningful knowledge in the KB. However, this process is very time consuming and extremely difficult when compared to cascading deletion in relational databases, because entities in the KB have complex relationships with many others. To ensure the consistency of the knowledge base, all relationships have to be examined carefully before deleting. Let us re-examine the example above. After the peoples presence in the SEMSRoom has been identied, the information about phones in the SEMSRoom might no longer be needed. However, when we delete this information, the KB now lacks the knowledge about the devices locations.

Lets examine a post meeting scenario, when people have left the SEMSRoom. Tom issues the query “where is my phone”. The system traces back in history to see where the phone was last seen. This may lead to unknown or an incorrect answer.

We note that, with deletion of context, it is not necessary when there is adequate data storage and also since, context information are timestamped. In fact, we typically do not delete older context information since they can be used and reused for historical queries. However, we can add rules to the knowledge base that would replace low-level context with an inferred higher level context. Recent theories on forgetting in knowledge bases and logic programming can be explored for a semantically grounded way of deleting or ignoring older context knowledge (Wang, Wang, Topor, & Pan, 2010).

The third layer is the **community layer**. It includes six different types of agents that represent a PBVC: Community Agent, Place Agent, Contributed-Context Agent, Device Agent, Service Agent and Monitoring Agent. The Community Agent acts as a “receptionist” or concierge for participants of the community. It knows other agents in the community. Whenever a request is sent to the community, this agent receives and finds a suitable agent for that task and introduces them to each other. The Place Agent is an agent that, based on the user's current location (e.g., GPS coordinates), maps raw positioning information to meaningful place information. Context information is gathered and processed via sensors[11] in the Context Processing Layer and Context Gathering Layer.

The top layer is the **application layer**, containing the User Agent (UA) and the Mobile User Agent (MUA). UA and MUA allow the user to communicate with other software agents in the lower layer. It also has the ability to communicate and to collaborate with other user agents in different applications. The MUA is built for running on mobile devices, such as Nokia N95, etc. The UA is built for running on resource-rich devices, such as laptops or desktops.

5. Applications using PlaceComm

In this section, we first illustrate the support for place discovery and community-based trust queries in our framework, and then describe applications that have been built based on the PlaceComm framework. All of the applications described in this section have been published via conferences, workshops and journal articles (Nguyen et al., 2007; Nguyen et al., 2008; Nguyen, Loke, Torabi, & Lu, 2009; Nguyen et al., 2009; Nguyen et al., 2010; Nguyen et al., 2010). However, here, we briefly summarize three of these applications and note how they are built on the PlaceComm framework.

5.1. Support for place discovery and community-based trust-related queries

5.1.1. Place discovery

In this section, we describe how we implemented a notion of place discovery.

5.1.2. Place by location

First of all, place is identified by a postal address. For retrieving an address, we used the Google Maps Reverse Geocoding web service.

Place by tagging: a place can be identified by significant objects, such as a GSM tower, a wireless access point or even an electronic tag. This method is integrated into PlaceSense and the Semantic PlaceBrowser, so that once a user finds a special object around, he/she sends Query 1 to discover or detect a place.

Query 1. Place by Significant Object (GSM Tower)

```
SELECT DISTINCT ?plc
WHERE { ?obj rdf:type:Object;
   :ObjectID?OjbID;:belongToPlace?plc.
FILTER (?OjbID="TowerName"^^xsd:string)}
```

For smaller scale places such as room scale, we can predefine a place and then attach a tag to represent the place. A tag is represented as a device in the PBVC ontology and it can be Bluetooth ID, RFID or even a Wireless Access Point. In the PlaceSense application (Nguyen et al., 2009), we apply this method to identify the post-graduates' office, a post office and a seminar room. Query 2 extracts the place name, based on the tag.

Query 2. Place tagging

```
SELECT DISTINCT ?plc
WHERE { ?device rdf:type:Device;
   :deviceID?devID;:belongToPlace?plc.
FILTER(?devID="SensedID"^^xsd:string)}
```

Place by people: by examining the KB, we found that there are some places where a group of people usually gathers at a specific place and time. For example, Query 3 lists places where Tuan's friends usually gather.

Query 3. Place of my friends

```
SELECT DISTINCT ?plc?timestamp
WHERE { ?person rdf:type:Person;
   :presenceAtPlace?plc;:friendOf?mine.
FILTER (?mine=:TuanNguyen)}
```

Place by event: also from examining the KB, Query 4 gives an answer, that is, the most crowded place at the moment. We envision that if each place has its own query-able knowledge base, we can identify the place density. This approach is inspired by Calabrese and Ratti (2007), where the authors examined the GSM signal at a stadium, where Madonna's concert had been organized.

Query 4. Place that has many people present

```
SELECT DISTINCT ?plcName?num
WHERE { ?vPlace rdf:type:Place.
?vPlace:placeName?plcName.
?vPlace:numberOfPeople?num.
OPTIONAL { ?crowder rdf:type:Place.
?crowder:numberOfPeople?cnum.
FILTER (?cnum >?num)}
FILTER (!bound (?crowder))}
```

Query 5 shows a fairly strong attachment of event and event attendees to a place.

Query 5. List device appears in two places

```
SELECT ?device?place?person
WHERE { ?device rdf:type:Device;
   :hasOwner?person;
   :devicePresenceAtPlace?place.
  ?place rdf:type:Place; :placeName?pname.
FILTER ((?pname="FlemmingtonRaceCourse") ||
          (?pname="Sydney"))}
```

Place by Rhythm: by capturing the speeds of vehicles in the KB, we found that a place (e.g., a road segment) can be identified by the traffic of vehicles traveling on it. The variation of vehicles' speeds on that road is similar to a rhythm; in addition, that rhythm is not the same everyday, it is different according to time and days. This finding can also be related to the Place literature discussed in Cresswell (2004), when a place is not static, but is a process.

[11] We take a broad definition of sensor as any device that can provide useful context information.

Query 6. List average speed of all vehicle traveling through particular place

```
SELECT avg(?speed)
WHERE { ?subject rdf:type sam:Location;
  sam:belongToPlace?place; sam:speed?speed;
  sam:timestamp ?timestamp.
FILTER(((?timestamp>=T1)& & (?timestamp<=T2))
& & (regex(str(?place),"EasternFreeway")))}
```

Place "without location": we can take the co-present IDs into account and search for related people, such as any friend of mine, community members and any relatives. We cannot resolve the place to a location but we can find our friends co-present.

5.1.3. Context-aided trust

While we note that trust is a broad topic, the trust aiding mechanism illustrated here aims to use context history. We illustrate this idea via queries below that take context into account to aid trust.

Trust about other people or agents: once I receive an answer to my query from a person, I would like to know more about that person to decide whether to trust him/her or not. There are many ways to do this, and we chose the simplest way: if the person is well known in this community, the chance of the person cheating and giving wrong answers is lower; therefore we use a query about how many people in this community know him/her (Query 7). In addition, to make sure, we also issue another query about how many friends that he/she has (Query 8), and how long he/she has been in this place, in this community. The "older" person in the community is usually more reliable than the freshman. If a person within the community is known by most people, he/she can be considered reliable.

Query 7. How many people in community A know person X?

```
SELECT count(?people)
WHERE { ?people rdf:type:Person;
  :knows?thisPerson;:belongToCommunity?commID.
  ?thisPerson:personID?perID.
FILTER ((?perID="X")& &
(?commID="A"))}
```

Query 8. How many people in community A are friends of person X?

```
SELECT count(?people)
WHERE { ?people rdf:type :Person;
  :friendOf?thisPerson;
  :belongToCommunity?commID.
  ?thisPerson:personID?perID.
FILTER ((?perID="X")& & (?commID="A"))}
```

Trust about place: once a person has a phone, he/she will gain access to some restricted area, so the system may ask for obvious events that only the owner can know, such as "where have you been since last night or at a specific time in the past?" Because the KB records everything, it is easy to find this out. In addition, the query "are there any friends of mine, who bought an iPhone in this store?" Query 9 means if many friends of mine have also bought from this store, I can "trust" this store.

Query 9. List friends of mine, who bought an iPhone from the shops in Northland shopping center?

```
SELECT ?person?products?someShops
WHERE { ?person rdf:type:Person; :friendOf?mine.
  ?person:buyThing?products.
  ?products rdf:type:Device;
    :deviceName?dName;:isSoldAt?someShops.
  ?someShops:locatedAt?place.
FILTER ((?mine=:TuanNguyen)
  & & (?place=:Northland) & & regex
  (?dName,"iPhone"))}
```

Trust with knowledge about activities in the past and present: consider the query "tell me what events have you been involved in?" (Query 10). The general idea is to give users questions about what they have done in the past that can be retrieved by the system. If they answer correctly, they are trusted. For example, a telephone company can ask its customer about five recent calls to identify the person as the right owner of a phone.

Query 10. What activities are my friends involved in?

```
SELECT ?activities?place?item
WHERE { ?activities rdf:type:ShoppingActivity;
  :happenAt?place;
  :hasPersonInvolved?personInvolved;
  ?personInvolved foaf:friendOf?mine.
FILTER (?mine = foaf:TuanNguyen)}
```

In our prototype, the MUA is equipped with a tool for capturing QR Codes.[12] The QR code can be used to encode a string that represents a recent ID of a place. The point here is only if the person is in front of the restaurant, he/she can see this code. Furthermore, to ensure this, the QR code can be periodically dynamically changed to make sure that only a person present there can see it - anyone else too far from the restaurant cannot. Query 11 shows the process of finding an available table in the restaurant with the dependent QR code. After the user captured the code, he/she will send a query to reserve a table with that code. Periodically auto-generated fresh QR codes that are valid for a fixed period implies that the user is near the restaurant at that time.

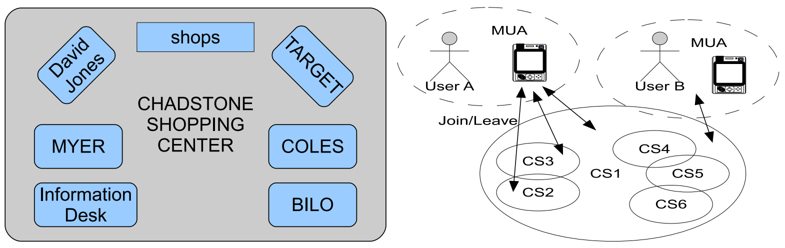
Query 11. Finding an available table query

```
SELECT ?empty_table
WHERE { ?empty_table rdf:type:Device;
:isAvailable?available.
?Captcha rdf:type:Device;:deviceID?captchaID.
FILTER (?available = YES)& &
     (?captchaID="DetectedID")}
```

5.1.4. Community stack: concept and prototype

As discussed in Sections 2 and 3, a physical place can be shared among many different communities. In each community, we have a set of services. When a user steps into a place that has more than

[12] QR Code Library, http://qrcode.sourceforge.jp.



(a) Chadston shopping center (b) The system model

Fig. 5. Scenario map and model.

one community, he/she logically belongs to multiple communities. In addition, he/she can be served by a set of services provided by different communities. This idea was first proposed in Nguyen et al. (2007). To illustrate the idea of a community stack, we consider the following scenario. When Tom has arrived from overseas to a new country, he starts to adapt to a new place. After he is picked up at the airport, he goes to a shopping center to buy the things he needs for a new life in a new place. Consider an application where a person, Tom, with three digital devices enters a community within a community (e.g., he enters the Myer department store community, after entering the shopping centre community (e.g., Chadston, Fig. 5(a))). He can travel around the shops, supermarket and finally leave.

There are three devices on the user - a smartphone, a watch, and a jacket with embedded computers inside, all connected in a Personal Area Network using Bluetooth technology. The smartphone can communicate with the environment via WiFi connectivity. The smartphone is the master device in the PAN. It is involved in managing other devices in the PAN, doing joining/leaving community activities. The smartphone can also receive advertising from shops. Consider our example scenario, when Tom is in the shopping center and he wants to buy 100 grams of cheddar cheese, the smartphone will create a "buying cheese" propose message and sends it to a (pre-configured) server, so that the shopping center software agent can help him find a merchant with the best price. The jacket receives a broadcast of trendy colors (assuming that the jacket can change colors) and digital images which it can display (as a fashion thing). The watch interacts with some servers of the community to synchronize time and downloads a list of events in the area adding them to its calendar. A shopping center can be modeled as in Fig. 5(b). In this example, there are five stores in the Chadstone community (CS1): the Myer (CS2), the David Jones (CS3), Target Supermarket (CS4), Coles supermarket (CS5) and BILO supermarket (CS6). This scenario illustrates the idea of communities within communities.

The architecture of the concept is shown in Fig. 6, which is a subset of the PlaceComm framework.

There are four types of agents in the PlaceComm framework which have been used in this application: the mobile user agent (MUA), the community agent, the service agent and the place agent. A possible scenario is as follows. The user, Tom, first goes to the Chadstone shopping center (CS1). In CS1, there are many community agents corresponding to inner communities within the Chadstone community such as Myer (CS2), David Jones (CS3), Target (CS4), Coles (CS5) and Bilo (CS6). The MUA on the smartphone performs the joining and leaving protocol whenever the user steps into or out of the community zones. Say, the user then goes into CS2, CS3, and stops in the middle of CS1. He then decides to buy food, cheddar cheese for example, and will be aided by services in the respective communities.

5.2. Student digital assistant

Improving on the prototype Community Stack, we produce an application called Student Digital Assistant (SDA), which is deployed in the university campus. The details of the work was first presented in Nguyen et al. (2008), but here, we relate it to the PlaceComm generic architecture. When Tom has arrived and enrolled in the University. He goes to the University's campus and starts finding places: the University's library, a bookshop, the East Lecture Theater (ELT) and the heart of the University; the Agora. In this scenario, when he is approaching the ELT, the software agent tells him that the lecture will be canceled. Fig. 7(a) shows the predefined area for each PBVC. We use four points to define the area to form a rectangle. Each point of the rectangle is determined by GPS coordinate data.

Therefore, when the user is in front of a particular place, the system can detect the user's location. For example, when the user is in location X in Fig. 7(a), the system will know that s/he is inside those PBVCs: the library, bookshop, the East Lecture Theatre (ELT) and the Agora. There are many ways to define an area for a PBVC. In our prototype, we use rectangles, but with more GPS points, n-sided polygons can be used. For the indoor system, we use iButton technology to detect a user's presence by assigning a person with a specific iButton ID (in this case, the iButton can be embedded in the user's ring, see Fig. 7(b)). The system architecture is shown in Fig. 8, which is depicted as a tailored version of the PlaceComm framework. We suppose that there are many services in a PBVC. The number of available services will increase if the user joins many PBVCs. The ServiceDisplayAgent and PBVC agents can also communicate with specific service agents to get specific service information.

5.3. Semantic place browser

The Semantic PlaceBrowser[13] exploits the strength of a Place Context Knowledge Base with the fundamental technology of PlaceSense and PlaceAware. The details of PlaceBrowser are in Nguyen et al. (2010).

The idea of browsing the environment interests many researchers (Castelli, Mamei, Rosi, & Zambonelli, 2006; Nakamura, Minakuchi, & Tanaka, 2006; Bainbridge, Jones, & Arter, 2007). However, little research has focused on sharing the captured context information between different applications. In addition, we exploit the context knowledge base, which is the main part of PlaceComm framework. We implemented 24 predefined queries for acquiring knowledge of the community. As an example, one SPARQL query that can be sent from the MUA to find where Tuan's friends are (within a large place) from time T1 to T2, as in Query 12.

[13] A demonstration of Semantic PlaceBrowser can be found at these links: http://tuannnguyen.blogspot.com/2011/04/semantic-placebrowser-understanding.html, https://sites.google.com/site/tuannguyenlatrobe/research/ running-semantic-placebrowser.

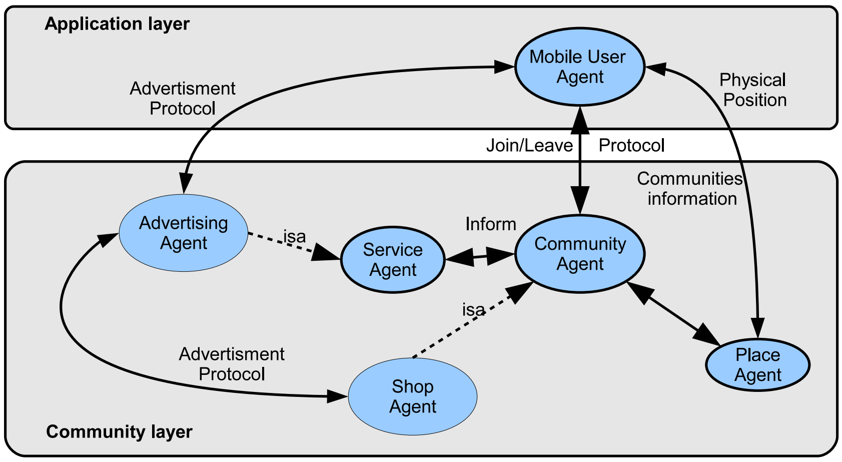


Fig. 6. Community Stack architecture, a specialization of the PlaceComm architecture.

Query 12. Where are Tuan's friends?

```
SELECT ?location ?person
WHERE { ?location rdf:type plc:Location.
  ?location pbvc:hasTimeInstant?timestamp.
  ?timestamp pbvc:timestamp ?value.
  ?location pbvc:belongtoPersonalContext?PersonalContext.
  ?PersonalContext pbvc:createdByPerson?person.
  ?person foaf:friendOf?personTuan.
  ?personTuan foaf:name?nameOfTuan
FILTER ((?value >=T1 &&?value <= T2) && (?nameOfTuan=
   ‘‘TuanNguyen"
  ^^xsd:string))}
```

With the help of a high level query language, we can list services that are relevant to the user's current interest (Query 13).

Query 13. List all services at all places that match with Tuan's current personal interest

```
SELECT ?service,?place,?category
WHERE { ?service rdf:type:Service;
     :serviceAtPlace?place;
     :serviceCategory?category.
FILTER (?category IN
    ( SELECT ?currentPersonalInterest
    FROM ?PersonalContext rdf:type:PersonalContext;
       :currentPersonalInterest?
         currentPersonalInterest;
       pbvc:createdByPerson?person
    WHERE ?person=:TuanNguyen))}
```

6. Performance evaluation of the PlaceComm framework

In this section, we detail an evaluation and discuss the challenge of building context-aware applications with regard to performance.

6.1. Performance evaluation setup

We envision that when the PlaceComm framework is put to use in reality, the size of the KB will depend on usage. But in order to see how a basic version of the PlaceComm implementation would actually respond to users' queries, from a small scale to a large scale, we prepared different knowledge bases, with sizes ranging from thousands to a million of instances. Table 1 shows different KBs with different sizes. KB0 is the original data set, in which instances are real data collected during our experiments in Nguyen et al. (2009), Nguyen et al. (2009) and Nguyen et al. (2010).

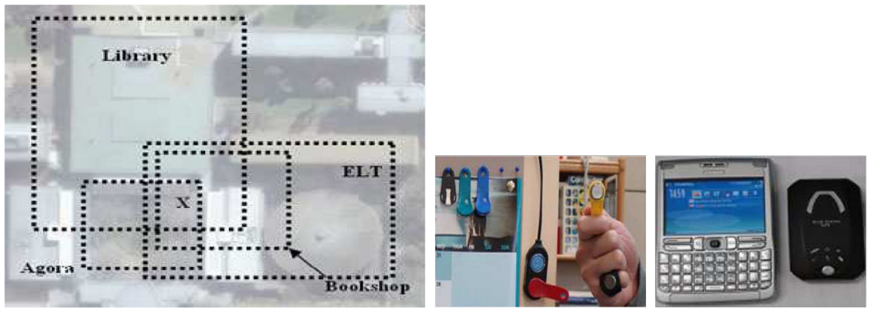
Listing 7. Snapshot of an instance.

```
<Location rdf:ID="Location1280203411796gen1283064650721">
  <timestamp rdf:datatype="#long">1246060503581</timestamp>
 <speed rdf:datatype="#float">91.231</speed>
 <hasPostalAddress rdf:resource="#Eastern_Fwy3108"/>
 <lon rdf:datatype="#float">145.12946</lon>
 <alt rdf:datatype="#float">67.45</alt>
 <date rdf:datatype="#string">Sat Jun 27 09:55:03 GMT+10:00 2009</
     date>
 <lat rdf:datatype="#float">-37.796993</lat>
</Location>
```

For data in KB2 to KB10, we generate different ±10% speed information for location from real data. Listing 7 shows a location instance generated from real data. This is used for simulation of the vehicles' speed on specific roads. All KB are stored in MySQL database servers using ProtegeOwl API for persistent storage. The evaluation server has hardware specifications: Intel Core 2 Quad CPU Q9300, 2.5 GHz, 8 GB RAM, 1TR Byte HDD, 6 GB Java Heapsize and Opensuse 64 bit operating system. Table 1 shows different KBs names and their sizes.

6.2. Evaluation queries

We used eight sample queries (Q14 to Q21) to evaluate the system. The purpose of this evaluation is to study the efficiency of typical place service queries on the place KB in order to study the feasibility of our concept of PBVC with a centralized KB. The queries are represented in the SPARQL language. Query 10 lists all members of the La Trobe Vietnamese postgraduate student association.



(a) The PBVCs for outdoor places (b) The iButtons (c) The smart-phone and GPS receiver

Fig. 7. SDA running site and devices.

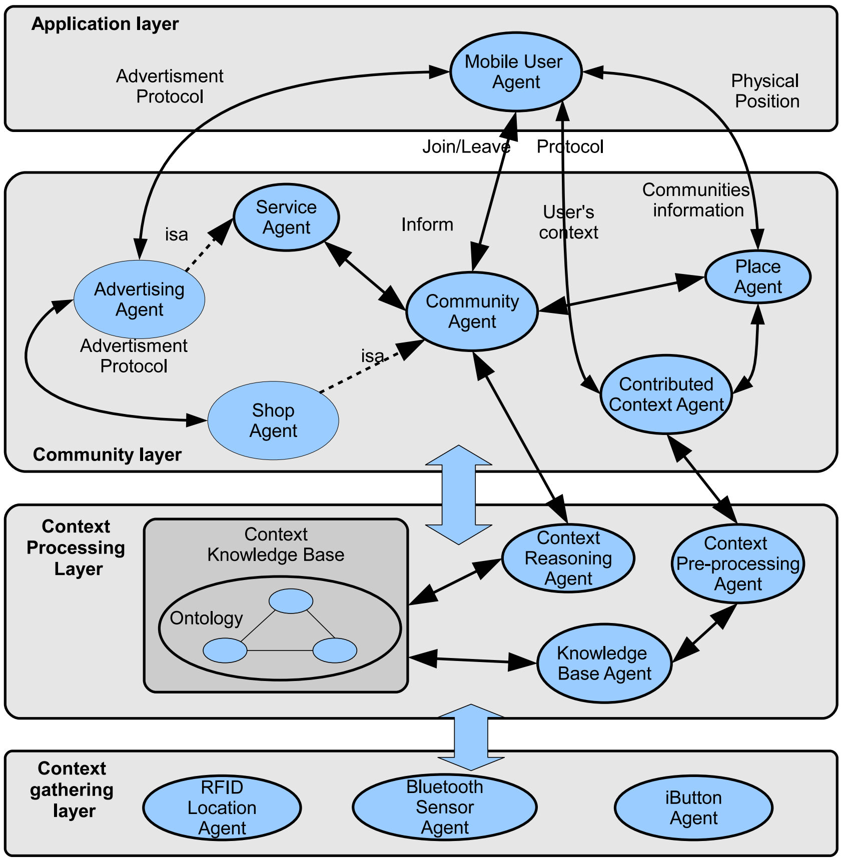


Fig. 8. The system architecture of SDA, which is a tailored version of the PlaceComm architecture.

Query 14. List members and place of La Trobe Vietnamese postgraduate students

```
SELECT?people?plc
WHERE {?pbvc rdf:type:PBVC;
  :hasMember?people; :hasPlace?plc.
FILTER (?pbvc =:LaTrobeVNPostGradStudent)}
```

Table 1
Different knowledge bases with different sizes (number of instances).

	KB0	KB2	KB4	KB6	KB8	KB10
Instances (×1000)	10	200	400	600	800	1000

Query 15 lists speeds of all vehicles on the Eastern Freeway; the segment of this freeway is extended to four suburbs (post codes 3104, 3108, 3109, 3130).

Query 15. List all speed of vehicles on Eastern Freeway

```
SELECT?speed
WHERE {?loc rdf:type:Location;
:speed?speed; :hasPostalAddress?pAddress.
FILTER ((?pAddress=:Eastern_Fwy3104) || (?pAddress=
  :Eastern_Fwy3108)
    || (?pAddress=:Eastern_Fwy3109) || (?pAddress=
      :Eastern_Fwy3130))}
```

Query 16 is to find whether any friends of TuanNguyen attended the event called “HelloVietnam”.

Query 16. List all Tuan’s friends, who attended the HelloVietnam event

```
SELECT?event?people
WHERE {?event rdf:type:Event;
   :hasPersonInvolved?people.
?people rdf:type:Person; :friendOf?mine.
FILTER ((?mine=:TuanNguyen) & &
  (?event=:HelloVietnam))}
```

Query 17 simply lists all locations contributed by users in the knowledge base. Note that this query gives the largest number of results, because the location context is the easiest to get. In addition, from the KB2 to KB10, this information is generated to test the performance of the KB. Query 18 browses the devices that belong to Tuan’s friends in Room 219, which is the postgraduates’ shared office.

Query 17. List all locations in the KB

```
SELECT?loc
WHERE {?loc rdf:type:Location.}
```

Query 18. List all Tuan’s friends devices

```
SELECT?dev?people?plc
WHERE {?dev rdf:type:Device;
     :devicePresenceAtPlace?plc;
     :devicePresenceAtPlace
   ?timestamp;:hasOwner?people.
   ?people rdf:type:Person; :friendOf?mine.
FILTER ((?mine=:TuanNguyen) & & (?plc=:PS1R219))}
```

Query 19 checks whether any friends of Tuan’s are at the Melbourne Cup’s place.

Query 19. List people who are friend of TuanNguyen at MelbourneCup2008 place

```
SELECT?people?plc
WHERE {?plc rdf:type:Place; :hasDevicePresence?dev.
?dev rdf:type:Device;:hasOwner?people.
?people rdf:type:Person;:friendOf?mine.
FILTER ((?plc =:MelbourneCup2008) & &
  (?mine=:TuanNguyen))}
```

Query 20 lists all devices which have joined the Melbourne Cup’s PBVC.

Query 20. Show all device at Melbourne Cup 2008 place

```
SELECT?devName?pbvc
WHERE {
?pbvc rdf:type:PBVC;:hasMember?dev;:hasPlace?plc.
?dev rdf:type:Device;:deviceName?devName.
FILTER (?plc=:MelbourneCup2008)}
```

Query 21 lists all locations that have a postal address. This query returns many results. However, the result set is smaller than Query 17 because some locations cannot resolve a postal address.

Query 21. Lists all locations that have a postal address

```
SELECT?addr?loc
WHERE {?loc rdf:type:Location;
  :hasPostalAddress?addr.}
```

6.3. Evaluation results

The process of evaluation is described as follows. First, we run the PlaceComm console to connect to the KB. The predefined queries are loaded into the console to prepare for the run of the performance tests. The text area shows the results of the run. Each result is an RDF triple. To better understand the results, we show the number of results (i.e., number of items satisfying the query) in Table 2. We can see that the number of results are the same for queries Q14, Q16, Q18, Q19 and Q20. The sizes of result sets are increasing in Q15, Q17, and Q21, because those queries are about location information, which are generated in the KB. Q17 has the most results. Note that for some queries, even when the KB is larger, the result set might remain the same size, clearly.

In this evaluation, we measure two factors: (i) the average time taken for each query in each KB (just to retrieve a handler to the potential result set); and (ii) the average time taken for computing the result set for each query (e.g., when we traverse the semantic links to obtain results). We first measured the average (over ten trials) time to get a result handler for each query (from Q14 to Q21) in each KB - according to the SPARQL java API, it just returns the handler for the ResultSet variable. We found that this response time is roughly equal (300 ms), no matter how large the KB is. Then, for the actual time to compute the individual result sets, it is shown in Figs. 9 and 10. We can see from the figures, generally, that the larger the KB, the longer it takes to compute/traverse. However, Q15 gives the longest compute/traversal times though with a far smaller result set than Q17 and Q21 (taking about 18 s to 44 min to complete, for different sizes of result sets), due to the complexity of the query. Similarly, Q18 takes far longer than Q14, Q16, Q19

Table 2
Size of result set for each query in each KB.

	KB0	KB2	KB4	KB6	KB8	KB10
Q14	8	8	8	8	8	8
Q15	129	3451	6807	7599	11619	17840
Q16	4	4	4	4	4	4
Q17	8796	198314	398318	598321	798326	998665
Q18	5	5	5	5	5	5
Q19	2	2	2	2	2	2
Q20	129	129	129	129	129	129
Q21	2458	55132	112552	164725	218948	279864

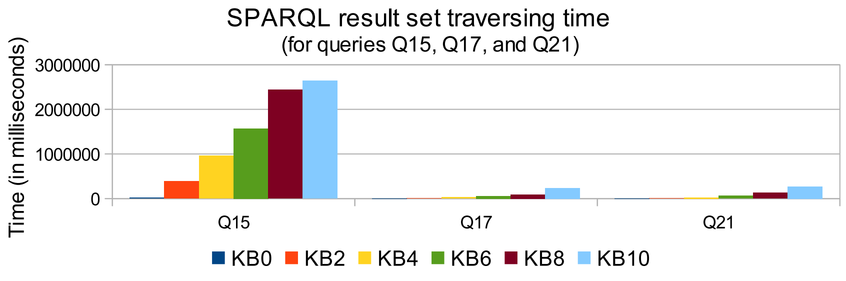


Fig. 9. The time queries Q15, Q17 and Q21 take to compute all results.

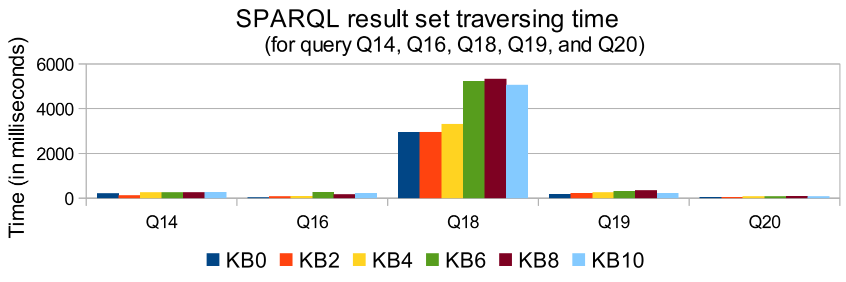


Fig. 10. The time queries Q14, Q16, Q18, Q19 and Q20 take to compute all results.

and Q20 for its result set to be computed due to its complexity, though it has fewer results than Q14 and Q20.

6.4. Discussion

The PlaceComm ontology can efficiently support large scale Knowledge Bases, which can contain up to millions of instances. The query response time can be improved via better hardware, e.g. if we have a higher hard drive speed. Ontology reasoning has been considered time consuming computationally, especially when the ontology and the knowledge become large. Therefore, to avoid this situation, we can reduce the size of the ontology/KB to a minimum, which is just adequate enough to model the environment. In addition, instead of building a big knowledge base in many places, we can build a KB for each place with a manageable size i.e., break into many smaller sites instead of one monolithic KB for all places. Our PBVC ontology is adequate for many applications as shown in Section 5 and yet it is not so complex that typical queries (for our applications) to a KB based on the ontology (with large numbers of instances) become intractable. We have not done optimizations for our results, which, hence, can be considered upper bound times. Also, we note that the complexity of the query also affects the response time. For example, Query 18 has a small number of results, but the time it takes increases with larger KBs.

7. Conclusion and future work

The notion of place should be central when building mobile services, and we believe this has not received adequate emphasis so far. On this note, we presented the idea of basing place-specific services on a place ontology. A key distinguishing feature of the PlaceComm framework is a common context knowledge base, shared among applications and contributed by agents. This makes the context more reusable, and context contributed in one service may be used in other services. In addition, the PlaceComm framework has demonstrated its versatility via a range of context-aware applications. A key contribution of this paper is to successfully combine agent, and ontological technologies, in order to enhance context-awareness services. This paper shows that practical performance can be obtained with such technologies, providing beneficial services in the way we have demonstrated in this paper.

Future work will look into performance improvements for much larger datasets with many concurrent users, more complex rule inferencing (but maintaining good performance), and scaling data storage via elastic cloud resources as accumulated context knowledge grows. One can look into extending the ontology to model particular types of places (e.g., hospitals, shopping malls, etc.). Also, we will extend this activity ontology with ideas from activity-based computing.[14]

References

Bainbridge, D., Jones, M., & Arter, D. (2007). A map-based place-browser for a PDA, Technical report, University of Waikato, Dept. of Computer Science, New Zealand.
Baldauf, M., Dustdar, S., & Rosenberg, F. (2007). A survey on context-aware systems. *International Journal of Ad Hoc and Ubiquitous Computing, 2*(4), 263–277.
Bellifemine, F., Poggi, A., & Rimassa, G. (2001). Jade: A fipa 2000 compliant agent development environment. In *AGENTS '01: Proceedings of the fifth international conference on autonomous agents* (pp. 216–217). NY, USA: ACM.
Bettini, C., Brdiczka, O., Henricksen, K., Indulska, J., Nicklas, D., Ranganathan, A., et al. (2010). A survey of context modelling and reasoning techniques. *Pervasive and Mobile Computing, 6*(2), 161–180.
Blackstock, M., Lea, R., & Krasic, C. (2007). Managing an integrated ubicomp environment using ontologies and reasoning. In *Proceedings of the fifth IEEE international conference on pervasive computing and communications workshops* (pp. 45–52). USA: IEEE Computer Society.
Calabrese, F., Kloeckl, K., & Ratti, C. (2009). *Handbook of research on urban informatics: The practice and promise of the real-time city. Chapter wikicity: Real-time location-sensitive tools for the city*. IGI Global.
Calabrese, F., & Ratti, C. (2007). Real time Rome. *Networks and Communication Studies – Official Journal of the IGU's Geography of Information Society Commission, 20*(3), 247–258.
Castelli, G., Mamei, M., Rosi, A., & Zambonelli, F. (2006). Browsing the world: Bridging pervasive computing and the web. In *International workshop on ubiquitous information systems* (pp. 7). Germany: Münster.
Chen, H., Perich, F., Finin, T., & Joshi, A. (2004). Soupa: Standard ontology for ubiquitous and pervasive applications. In International conference on mobile and ubiquitous systems: Networking and services (pp. 258–267).
Ciolfi, L. (2004). Understanding spaces as places: extending interaction design paradigms. *Cognition, Technology & Work, 6*(1), 37–40.

[14] http://activity-based-computing.org/.

Ciolfi, L., & Bannon, L. (2005). Spaces, spatiality and technology. *Chapter Space, Place and the Design of Technologically-Enhanced Physical Environments*, 217–232.

Ciolfi, L., Fitzpatrick, G., & Bannon, L. (2007). Settings for collaboration: The role of place. *Computer Supported Cooperative Work (CSCW), 17*, 91–96.

Cresswell, T. (2004). *Place: A short introduction*. Wiley-Blackwell.

Dey, Anind K. (2001). Understanding and using context. *Personal Ubiquitous Computing, 5*(1), 4–7.

Endres, C., Butz, A., & MacWilliams, A. (2005). A survey of software infrastructures and frameworks for ubiquitous computing. *Mobile Information Systems, 1*(1), 41–80.

Gómez-Pérez, A., Juristo, N., & Pazos, J. (1995). Evaluation and assessment of the knowledge sharing technology. In *Towards very large knowledge bases* (pp. 289–296). Holanda: IOS Press.

Gordon, E., & Silva, A. S. (2011). *Net locality: Why location matters in a networked world*. Wiley-Blackwell.

Wang, X. H., Zhang, D. Q., Gu, T., & Pung, H. K. (2004). Ontology based context modeling and reasoning using owl. In *IEEE international conference on pervasive computing and communications workshops* (pp. 18–22).

Hightower, J., Consolvo, S., Lamarca, A., Smith, I., & Hughes, J. (2005). Learning and recognizing the places we go. In *UbiComp* (pp. 159–176).

Hightower, J., LaMarca, A., & Smith, I. E. (2006). Practical lessons from place lab. *IEEE Pervasive Computing, 5*(3), 32–39.

Hill, Ernest F. (2003). *Jess in action: Java rule-based systems*. Greenwich, CT, USA: Manning Pub. Co..

Hobbs, Jerry R., & Pan, F. (2004). An ontology of time for the semantic web. *ACM Transactions on Asian Language Information Processing, 3*(1), 66–85.

Loke, Seng Wai (2002). Modelling service-providing location-based e-communities and the impact of user mobility. In *DCW '02: The forth international workshop on distributed communities on the web* (pp. 266–277). London, UK: Springer-Verlag.

Martin, D., Burstein, M., Mcdermott, D., Mcilraith, S., Paolucci, M., Sycara, K., et al. (2007). Bringing semantics to web services with owl-s. *World Wide Web, 10*(3), 243–277.

Moore, P., Hu, B., & Wan, J. (2008). Smart-context: A context ontology for pervasive mobile computing. *The Computer Journal, 53*(2), 191–207.

Nakamura, S., Minakuchi, M., & Tanaka, K. (2006). Ambientbrowser: Web browser in everyday life. *Ambient intelligence in everyday life* (Vol. 3864/2006, pp. 157–177). Berlin/Heidelberg: Springer.

Nguyen, T. (2008). Place-based virtual community – a new approach for context-aware applications. In R. Mayrhofer, A. Quiley, J. Kay, G. Kortuem, S. Ardon, & E. Rukzio, et al. (Eds.), *Advances in pervasive computing. Adjunct proceedings of the sixth international conference on pervasive computing* (pp. 148–153). Sydney, Australia: Austrian Computer Society. 987-3-85403-236-6.

Nguyen, T., Loke, Seng W., Torabi, T., Lu, H. (2009). Placeaware: A tool for enhancing social interactions in urban places. In *Proceeding of the 10th international symposium on pervasive systems, algorithms, and networks (ISPAN)* (pp. 143–147).

Nguyen, T., Loke, Seng W., Torabi, T., & Lu, H. (2009). Placesense: A tool for sensing communities. In *ISWPC'09: Proceeding of the fourth international symposium on wireless pervasive computing* (pp. 1–5).

Nguyen, T., Loke, Seng W., Torabi, T., & Lu, H. (2010). Making places legible: Ontology support for context-aware applications in place-based virtual communities. In *Ant 2010 – the international conference on ambient systems, networks and technologies, Paris, France* (pp. 73–80).

Nguyen, T., Loke, Seng W., Torabi, T., & Lu, H. (2010). Semantic placebrowser: A tool for place scale computing. In *Ubicomp in the large, pervasive 2010 workshop* (pp. 5–8).

Nguyen, T., Loke, Seng W., & Torabi, T. (2007). The community stack: Concept and prototype. *Proceedings of AINAW'07* (Vol. 2, pp. 52–58). Los Alamitos, CA, USA: IEEE Computer Society.

Nguyen, T., Loke, Seng W., Torabi, T., & Lu, H. (2008). Multiagent place-based virtual communities for pervasive computing. In *Proceedings of the sixth annual IEEE international conference on pervasive computing and communications* (pp. 602–608). Los Alamitos, CA, USA: IEEE Computer Society.

Nguyen, T., Loke, Seng W., Torabi, T., & Lu, H. (2011). A framework for context-aware applications in place-based virtual communities. *Journal of Ambient Intelligence and Smart Environments (JAISE), 3*(1), 51–64.

Niu, W., Kay, J. (2008). Location conflict resolution with an ontology. In *Pervasive* (pp. 162–179).

Niu, William T., & Kay, J. (2010). Personaf: framework for personalised ontological reasoning in pervasive computing. *User Modeling and User-Adapted Interaction, 20*, 1–40.

Quelch, John A., & Jocz, Katherine E. (2012). *All business is local: Why place matters more than ever in a global, virtual world*. Portfolio Hardcover.

Raz, D., Juhola, Arto T., Serrat-Fernandez, J., & Galis, A. (2006). *Fast and efficient context-aware services*. Wiley.

Sheng, Quan Z., Yu, J., & Dustdar, S. (2010). *Enabling context-aware web services: methods, architectures, and technologies* (1st ed.). Chapman & Hall/CRC.

Toye, E., Sharp, R., Madhavapeddy, A., & Scott, D. (2005). Using smart phones to access site-specific services. *IEEE Pervasive Computing, 4*(2), 60–66.

Wang, Z., Wang, K., Topor, Rodney W., & Pan, Jeff Z. (2010). Forgetting for knowledge bases in DL-Lite. *Annals of Mathematics and Artificial Intelligence, 58*(1–2), 117–151.

Ye, J., Coyle, L., Dobson, S., & Nixon, P. (2007). Ontology-based models in pervasive computing systems. *The Knowledge Engineering Review, 22*(4), 315–347.