Semantic Space: A Semantic Web-Based Infrastructure for Smart Spaces

Draft for IEEE Pervasive Computing

Xiaohang Wang, Daqing Zhang, Jin Song Dong, Chunyau Chin

xwang@i2r.a-star.edu.sg ...

Abstract. Semantic Space is a pervasive computing infrastructure that exploits the use of Semantic Web technologies to support explicit representation, expressive querying and flexible reasoning of contexts in smart spaces.

1 Introduction

Recent activity in pervasive computing research is attempting to merge the physical and digital worlds by incorporating physical and computing entities into smart spaces. Smart spaces consist of physical spaces (home, workplace, classroom, vehicle, etc.), extensively equipped with embedded sensors, augmented appliances, stationary computers, and mobile handheld devices. Applications in such environments need to be context-aware so that they can adapt themselves to rapidly changing situation[1, 2]. The dynamic nature of smart spaces creates great challenges for developing context-aware applications. To make these challenges concrete, we give a motivating scenario and highlight the research issues that are raised.

> Rachael wishes to contact her friend Joey, so she instructs her mobile phone to arrange a call. Upon requested, Joey's mobile phone checks the calendar and realizes that he is currently attending a seminar. The phone determines on his behalf that he should not be interrupted and schedules a call back to the time the seminar ends.
>
> Soon after the seminar, Joey is asked by Professor Geller to have a discussion in his office. Before the phone reminds Joey of the missed call as scheduled earlier, it wants to know whether his current situation is suitable for making the call. Based on various contexts (e.g., Where are you? Who are you with? What is the noise level? Is the door open or closed?) gathered by the smart space, the phone infers that Joey is in a conversation with his supervisor and then decides to re-arrange the call until he is available. A few minutes later when the conversation ends and Joey leaves the office, the phone finally reminds him of the missed call.

The realization of smart spaces relies on many different technologies. We chose to narrow our focus on following issues.

- *Explicit Representation*: Raw context data, obtained from a wide variety of sources, are likely to be available in heterogonous formats. Consequently, applications without prior knowledge of the representation of contexts are not able to make use of them. Therefore an interoperable smart space requires the meanings (or semantics) of contexts to be explicitly represented in the way independently-developed applications can easily understand.
- *Context Querying*: A smart space maintains a large amount of contexts and applications may need to selectively access a subset of them. Hence the smart space should be able to answer expressive queries about contexts (e.g., Who is in the same room together with Joey? When will the ongoing seminar where the user is either as an attendee or a speaker end?")

- *Context Reasoning*: Higher-level contexts (e.g., What is the user doing? What is the activity in the room?) are useful to context-aware applications as they provide summary descriptions about the state of users and surroundings. While sensors are not able to recognize such contexts, they need to be inferred from basic sensed contexts.

To address the foregoing issues, we developed a context infrastructure called Semantic Space that supports explicit representation, expressive querying and flexible reasoning of contexts in smart spaces. Our approach is inspired by the Semantic Web, the next generation of World Wide Web on which content is given well-defined meaning, better enabling computers and people to work in cooperation[3]. Based on a set of standards (e.g., RDF[4], OWL[5]) for representing machine-interpretable information, Semantic Web introduces a federated approach to knowledge management and information processing. In this paper, we will describe our approach to explore the use of these technologies in building a pervasive computing infrastructure that presents the following features:

- An ontology-based context model that provides the basis for explicit context representation and context interpretation in pervasive computing environments.
- A context infrastructure that allows contexts to be explicitly represented as semantic markups for easy interpretation by applications, enables applications to retrieve contexts using declarative queries, and supports the inference of higher-level contexts from basic sensed contexts

2 The Context Model

The presentation of contexts is an important part of pervasive computing environments. Context-aware applications adapt to changing situation on behalves of users, thus they need a detailed model about relevant aspects of users and surroundings, allowing them to share users' perceptions of the real world[6].

2.1 An Ontology Approach to Context Modeling

Within the domain of knowledge representation(KR), the term *ontology* is referred to as the formal explicit description of concepts, which are often conceived as a set of entities, relations, instances, functions and axioms[7]. Among Semantic Web standards, the Web Ontology Language is proposed to define and instantiate ontologies for Web information, thus enabling Web agents to exchange and interpret information based on a common vocabulary. The use of ontology to model contexts in pervasive computing environments introduces many advantages:

- OWL ontologies that represent contexts allow pervasive computing entities to share common understanding of the structure of contexts, thus enable applications to interpret contexts based on their semantics.
- The hierarchical structure of ontology helps developers reuse domain ontologies (e.g., ontologies of people, device and activity) in describing contexts, facilitating the building a practical context model without starting from scratch.
- Since contexts described in ontologies have explicit representations of semantics, context interpretation can be well supported by Semantic Web tools (i.e., federated query, reasoning and knowledge base). We can incorporate these tools into smart spaces to facilitate the management and interpretation of contexts.

2.2 Design the Context Model

Smart spaces cover a range of environment types such as homes, offices, workplaces, classroom, and vehicles. We do not aim to completely model all contexts in different types of smart spaces. Instead, we define an Upper-Level Context Ontology (ULCO) to provide a set of basic concepts that are common across different environments[8]. Among various contexts, we identify 3 classes of real-world objects (i.e., user, location, computing entity) and anther class of conceptual objects (i.e., activity) that

are most important to characterize smart spaces. These objects, linked with each other, form the skeleton of a contextual environment. They also can be used as primary indices into other associated contexts. For example, given a location, we can acquire related contexts such as noise, weather, the number of people inside, etc. Therefore, we choose to model these objects as top-level classes in ULCO.

One of the advantages of ontology is knowledge reuse[9]. We integrate a number of consensus domain ontologies, i.e., Friend-Of-A-Friend (FOAF[1]), RCAL Calendar[2] and FIPA Device Ontology[3], into ULCO to model contexts about users, activities and devices respectively. These well-defined ontologies provide a generic set of vocabularies that well suit the requirements of context ontologies.We only need to add additional properties that are useful to smart spaces. For example, FOAF only defines simple relationship between people (i.e., friendOf), but we extend it to support richer properties such as supervisorOf, studentOf, colleagueOf etc.

To keep the context model customizable to a particular smart space, it is intended to complement the classes defined in ULCO. In case that a new application needs additional classes that further specify the existing ones, they can be inherited from the classes of ULCO, forming a so-called Extended Context Ontology (ECO) (see Figure 1). In this way, developers can easily build detailed context models for newly-setup smart spaces. Moreover, the use of UCLO can support better interoperability between ECOs. Different ECOs will be able to interoperate by virtue of shared terms and definitions.

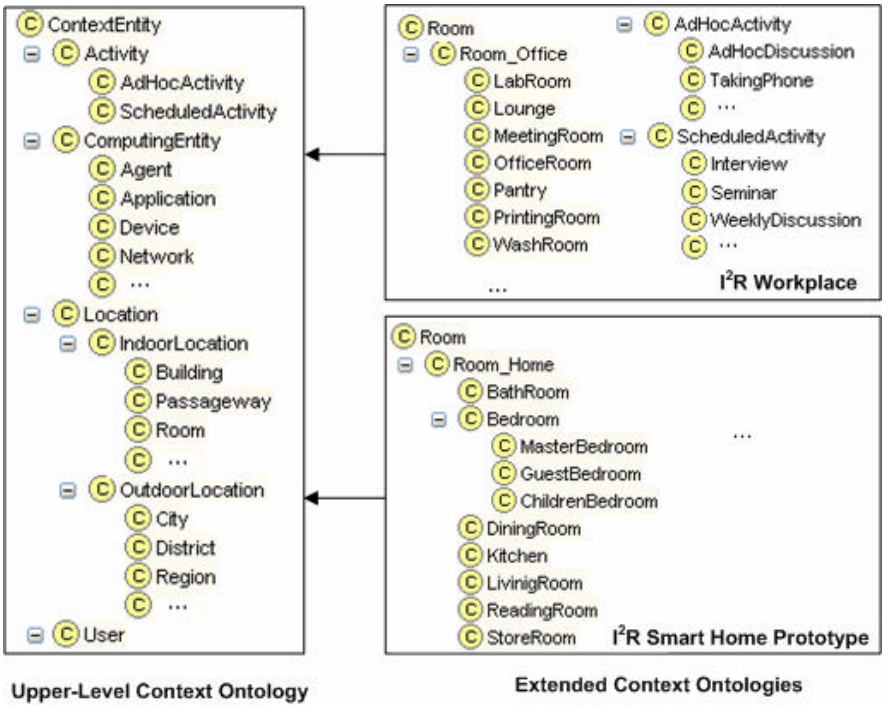

Fig. 1. Upper-Level Context Ontology and Extended Context Ontologies

2.3 Mark up Real-World Contexts

With our context model, contexts are represented as ontology instances and associated properties (so-called context markups) that can be easily interpreted by applications. Real-world contexts often originate from diverse sources, leading to dissimilar approaches to generating context markups. Let us take examples of the contexts involved in the smart phone scenario. Some of the contexts (e.g., name of a person, relationship between two persons, and scheduled time of a seminar) have relatively slow

[1] FOAF: http://xmlns.com/foaf/0.1
[2] RCAL: http://www.daml.ri.cmu.edu/Cal
[3] FIPA Device Ontology: http://www.fipa.org/specs/fipa00091/XC00091D.html

rates of change; they are often supplied by users. Markups of these contexts are usually generated by users. For example, we have a JavaScript application that allows users to online create profiles based on the ontology class User. Following example shows the context markup that describes RossGeller[4].

```
<User rdf:about="RossGeller">
    <name>Ross Geller</name>
    <mbox>ross@i2r.a-star.edu.sg</mbox>
    <homepage rdf:resource="www.i2r.a-star.edu.sg/~ross"/>
    <office rdf:resource="#Room209"/>
    <officePhone>1234</officePhone>
    <mobilePhone>6789</mobilePhone>
    <supervisorOf rdf:resource="#JoeyTribbiani"/>
    <! --More properties is not shown in this example-->
</User>
```

On the other hand, some other contexts (e.g., location, current time, noise level, door status) are usually provided by hardware or software sources. The marking up of these contexts needs to be performed by automated programs due to the high rates of change. Let consider the RFID indoor location system that tracks users' location by detecting the presence of body-worn tags. When RossGeller enters OfficeRoom209, the RFID sensor detects his presence and composes the context markup as described bellow.

```
<User rdf:about="#RossGeller"> <locatedIn rdf:about="#Room209"/> </User>
```

Since each OWL instance has a unique URI, context markups can link to external definitions through these URIs. For example, the URI http://www.i2r.a-star.edu.sg/SemanticSpace#RossGeller refers to the user defined earlier, and the URI http://www.i2r.a-star.edu.sg/SemanticSpace#Room209 refers to a room that is also defined elsewhere.

3 The Semantic Space Infrastructure

Semantic Space has a context infrastructure that allows contexts to be represented as semantic markups that applications can easily interpret. It enables applications to retrieve contexts using declarative queries, and support the inference of higher-level contexts from basic contexts. One such infrastructure is maintained in each smart space, which is often bound by the physical space in a non-restricted way. For example, two or more rooms may be joined to form a smart space, or a single room may be split into multiple smart spaces.

The context infrastructure consists of collaborating components (Figure 2). Context Wrappers obtain contexts from sources and dynamically generate context marks. Context Aggregator gathers context markups from Context Wrappers and asserts contexts into Context Knowledge Base (CKB). CKB links context markups into a single coherent model, and provides an interface for Context Query Engine to handle expressive queries from applications. Context Reasoner infers higher-level contexts from basic contexts in CKB based on application-specific rules.

3.1 Context Wrappers

Context Wrappers obtain raw contexts from sources and transform them to context markups. Contexts often originate from a wide variety of sources that could be either hardware sensors or software programs. We have a number of Context Wrappers that work with hardware sensors deployed in our prototypical smart space, including the Location Context Wrapper that has been described in Section 2.3, the Environment Context Wrapper that gathers environmental information (i.e., temperature, noise, light) from embedded sensors, and the Door Status Context Wrapper that reports the "open/close" status of the doors in each room. In addition, we also have several software-based Context Wrappers, including the Activity Context Wrapper that extracts schedule from Outlook Web Access, the Device

[4] "http://www.i2r.a-star.edu.sg/SemanticSpace#" is used as the default base namespace throughout this paper.

Context Wrapper that monitors the status of different networked devices (e.g., VoIP phone and mobile phone), the Application Context Wrapper that monitors the status(e.g., "idle", "busy", "closed") of different applications(e.g., JBuilder, Microsoft Word and RealPlayer) from their CPU usages, and the Weather Context Wrapper that acquires outdoor weather by periodically querying a Weather Web Service[5].

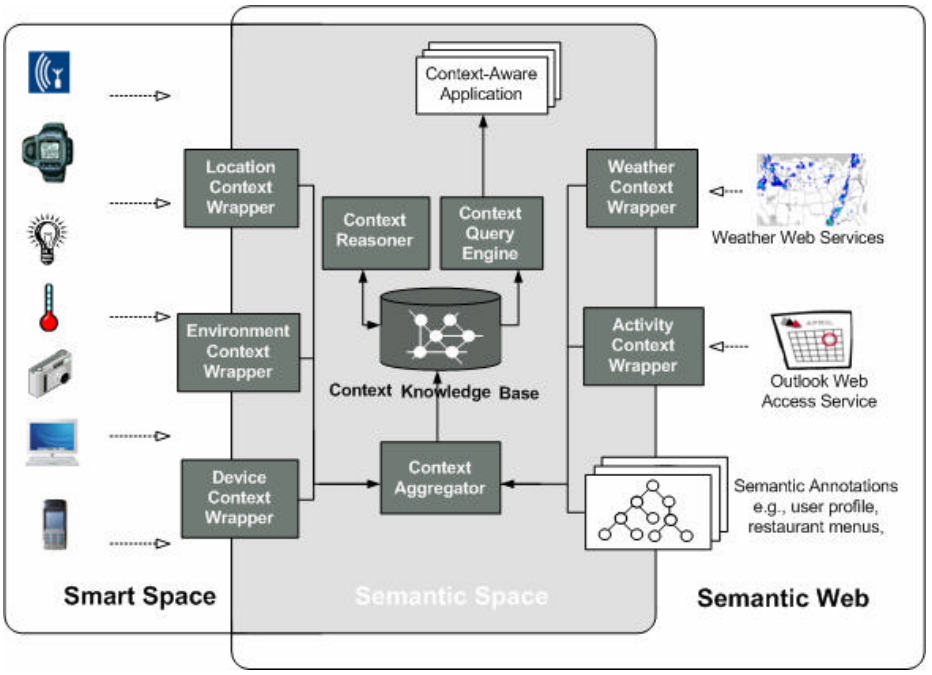

Fig. 2..Semantic Space Context Infrastructure

All Context Wrappers are self-configuring components that support a unified interface for getting contexts from sensors and providing context markups to consumers (i.e., applications and the Context Aggregator). Context Wrappers are implemented as UPnP[6] services that can dynamically join a smart space, obtain an IP address, and multicast its presence for others to discover. Context Wrappers uses UPnP General Event Notification Architecture (GENA) to publish the change of contexts as events that can be subscribed by consumers.

3.2 Context Aggregator

Context Aggregator discovers Context Wrappers and gathers context markups from them. Context Aggregator is implemented as an UPnP Control Point, inheriting the capability to discover Context Wrappers and subscribe to context events. Once a new Context Wrapper is attached to the smart space, Context Aggregator will discover it, register it in the service directory, and then obtains context markups from it. It asserts gathered context markups into CKB and also keeps updating it whenever a context event occurs.

3.3 Context Knowledge Base

Residing in each smart space, CKB provides a persistent storage for context knowledge. It stores the ECO for that particular space and the context markups that are either given by users or gathered from Context Wrappers. CKB links the context ontology and context markups into a single semantic model, and provides interfaces for Context Query Engine and Context Reasoner to manipulate correlated contexts.

[5] Weather Web Service: http://www.xmethods.com
[6] UPnP: http://www.upnp.org/

Contexts in smart spaces display very high change rates. CKB therefore always needs to be kept updated by Context Aggregator with fresh contexts. The scope of contexts managed by CKB also changes dependent on the availability of Context Wrappers. Developers may add a new Context Wrapper to expand the scope of contexts in a smart space; they also may remove an existing Context Wrapper when the contexts it provides are no longer needed. Context Aggregator is responsible for monitoring the availability of Context Wrappers and managing the scope of contexts in CKB. When a Context Wrapper joins the smart space, the provided contexts will be added to CKB. On the other hand, when a Context Wrapper leaves, the contexts it supplies will be deleted from CKB to avoid stale information.

3.4 Context Query Engine

Context Query Engine provides an abstract interface for applications to extract desired contexts from CKB. In order to support expressive queries, we adopt RDF Data Query Language (RDQL[10]) as the context query language. RDQL supports query over semantic models based on triple (i.e. <subject, predicate, object>) patterns. The use of query allows applications to selectively access contexts using declarative statements. Following shows a self-explanatory query statement for "when will the ongoing seminar where RossGeller is either as an attendee or a speaker be over?".

```
SELECT ?event, ?t2
WHERE (?event, <rdf:type>, <Seminar>),
      (?event, ?relation, <RossGeller>),
      (?event, <startDateTime>, ?t1),
      (?event, <endDateTime>, ?t2)
AND   (t1 < currentDateTime() && t2 > currentDaytime()) &&
      (?relation <eq> <attendee> || ?relation <eq> <speaker>)
```

3.5 Context Reasoners

Context Reasoner is the component that infers abstract, higher-level contexts from basic, sensed contexts. In Semantic Space, all contexts are explicitly represented in the way general-purpose reasoning engines can directly process, making it easy for developers to realize application-specific inferences simply by defining heuristic rules.

Applications may use application-specific rules that generate conflict results. Context Reasoner does not assert inferred contexts into CKB so that confliction in a single model can be avoided. Whenever the application needs certain higher-level contexts, it submits a set of rules to Context Reasoner. Context Reasoner, on the other hand, applies these rules to infer higher-level contexts on behalf of the application, and then returns newly inferred contexts without storing them in CKB.

Our current system uses Jena2 generic rule engine[11] to perform forward-chaining reasoning over CKB. Developers can write rules for a particular application based on its needs. The rules described bellow, related to the motivating scenario, illustrate how to infer the likely situation of users based on a variety of contexts about user(e.g., user's identity and his relationships with others), activity(e.g., activity's type and time interval), location(e.g., user's location, user's office location, others at the same location), and computing entity(e.g., status and ownerships). For example, the first rule examines whether a given person is currently engaged in a meeting, on the basis of location and schedule, i.e., if he is in the meeting location and current time (returned by currentDateTime()) is within the scheduled interval of the meeting, then he is likely to be at meeting.

```
type(?user, User), type(?event, Meeting), location(?event, ?room), locatedIn(?user, ?room), startDateTime(?event, ?t1),
endDateTime(?event, ?t2), lessThan(?t1, currentDateTime()),
greaterThan(?t2, currentDateTime())
=>situation(?user, AtMeeting)

type(?user, User), type(?phone, Phone), owner(?phone, ?user), status(?phone, ?busy)
=>situation(?user, TakingPhoneCall)

type(?user, User), type(?app, MicrosoftWord), registeredUser(?app, ?user), status(?app, ?busy)
=>situation(?user, AtWriting)
```

type(?user, User), type(?app, JBuilder), registeredUser(?app, ?user), status(?app, ?busy)
=>situation(?user, AtProgramming)

type(?user, User), locatedIn(?user, I2RCanteen), greaterThan(currentTime(), 12:00:00), lessThan(currentTime(), 13:30:00)
=>situation(?user, AtLunch)

type(?user, User), type(?room, Washroom), locatedIn(?user, Washroom)
=>situation(?user, UsingWashroom)

type(?user, User), studentOf(?user, ?user2), office(?user2, ?room), locatedIn(?user, ?room), locatedIn(?user2, ?room),
doorStatus(?room, closed), noiseLevel(?room, ?x), greaterThan(?x, 60)
=>situation(?user, MeetingSupervisor)

4 Implementation and Evaluation

4.1 Prototype Implementation

We have implemented our context infrastructure and set up our workplace in I^2R building as a testbed. We have defined the Upper-Level Context Ontology using OWL, based on which we built an Extended Context Ontology for our workplace. We implemented CKB, Context Reasoner and Context Query Engine using Jena2 Semantic Web Toolkit, and realized discovery and event notification mechanism using Siemens UPnP SDK v1.01. In addition to the general infrastructure, we also provide a number of concrete Context Wrappers (described earlier in Section 3.1.) that facilitate the sensing and markup of various contexts (i.e., location, schedule, temperature, noise, light, door status, device status, and application status). Figure 3a shows networked sensors and devices that we have integrated to provide contexts. Figure 3b shows the indoor location system that we developed using Ti-RFID Serial 2000.

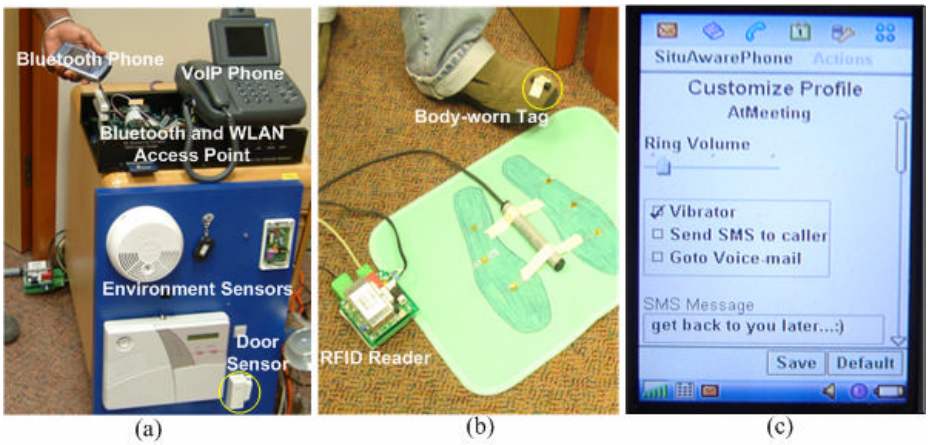

Fig. 3. (a) Networked sensors and devices; **(b)** RFID indoor location system; **(c)** Snapshot of **SituAwarePhone**: GUI for configuring the response mode in each situation.

4.2 Evaluation

Pervasive computing systems are difficult to evaluate because they often stress new functionality and usability over pure performance. Arguably, we believe the most important aspect of our infrastructure is what it enables; its features allows contexts to be represented as semantic markups that applications can easily interpret, enables applications to retrieve contexts using declarative queries, and support the inference of higher-level contexts from basic sensed contexts. Encapsulation of these features into

API helps developers to build applications that would otherwise be difficult to build. The evaluation consists of two parts; we first evaluate the infrastructure by building a real-world application using it, and then measure the system performance.

4.2.1 Application Development

The wide-spread use of mobile phone has raised many social problems, e.g., phones ring in meetings or important conversations. Users often have to change the settings of their phones according to the circumstances in order to avoid inappropriate usage,. However, frequent interactions with mobile phones impose signification user distractions. To solve this problem, we developed a context-aware application called **SituAwarePhone** that can adapt the mobile phone to the changing situation while minimizing user distractions.

One of the application scenarios has been described earlier in Section 1. When **SituAwarePhone** receives an incoming call, it will first infers user's situation using a set of rules (described in Section 3.5), and then adapts its response mode (e.g., adjust volume, set vibration, schedule a call back, send message, forward to voice mailbox) automatically. **SituAwarePhone** also queries the smart space for various contexts that help in adaptation, e.g., it asks for "the end time of the seminar the owner is engaged in"(described in Section 3.4) in order to schedule a call back, or takes account of "the identity of the caller" and "the identities of other people in conversation" to decide whether to let the call interrupt the conversation.

In a smart space, the context infrastructure supports the entire process of gathering contexts from sources, managing contexts using knowledge base, handling queries from applications and reasoning about contexts based on rules. The infrastructure provides a simple client-side API for applications to access its functionalities, hiding the complexity of underlying context processing. Following lists the API methods that applications can use to deal with contexts.

```
SmanticSpace(String ServerURL) throws InstantiationException;
//Instantiate a client object of the context infrastructure

RDFModel SemanticSpace.ContextQueryEngine(String Query) throws QueryException;
//Query contexts from the context infrastructure using the statement defined in Query

RDFModel SemantiSpace.ContextReasoner(String RuleSet) throws ReasoningException
//Request Context Reasoner to perform inference using the rules defined in RuleSet.
```

With the help of client-side API, we built **SituAwarePhone** on top of SonyEricsson P900 mobile phone (Figure 3c). The implementation amounts to approximately 800 lines of Java code, the majority of which deals with MIDlet GUI and different response modes. Only about 20 lines of code (including the code to import library, issue query, perform reasoning, parse returned model, and handle exceptions) on application side deal with contexts. From this example, we show how the context infrastructure can greatly ease the development of context-aware applications.

4.2.2 Performance

We evaluated the performance of the system by measuring the response time of context querying and reasoning. The experiment was conducted on a workstation with 2.4 GHz Pentium-4 CPU and 1.0 GB RAM running Redhat 9.0. We used five context datasets to test the scalability of our system. Among these datasets, the one with approximate 3000 triples (or 600 OWL classes and instances) is the real dataset used in our prototypical smart space, while the other four are synthetic datasets with unique classes and instances.

We used the complex query statement described in Section 3.4 to measure the performance. We varied the number of matched results from 1 to 10. The results are shown in Figure 4a. We expected the response time to be loosely proportional to the size of context dataset and the number of matched results, which the experiment corroborated.

We created four sets of rules to test the performance of context reasoning. We used the rule set with 10 rules used by **SituAwarePhone** (refer to Section 3.5), and created the others with increasing numbers of rules based on this one. From the result (Figure 4b) we can see that rule-based reasoning

is computationally intensive, and the response time largely depends on the size of dataset and the rule set applied.

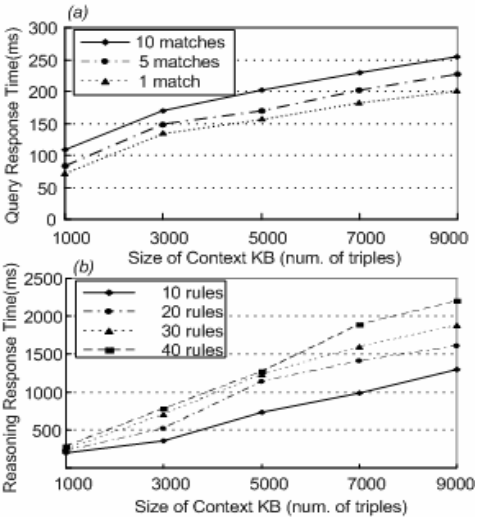

Fig. 4.**(a)** Query performance; **(b)** Reasoning performance

As the scale of the smart space increases, the response time of reasoning will be human perceivable. However, the performance evaluation suggests that ruled-based context reasoning work in practice for a useful set of applications in pervasive computing environments. Let us take the use of **SituAware-Phone** for example. Before the mobile phone responses to an incoming call, it needs to reason about the callee's situation. Context reasoning will cause a perceivable delay (about one second's time), which sometimes matters to users. For example, the callee may wish to answer the call from his supervisor as soon as possible, and the caller always expects a quick response. The feedback from our invited evaluators, both as callers and callees, has showed that the response time of **SituAwarePhone** is acceptable to most of them. Similarly, we believe the context infrastructure can well support a large number of potential applications (e.g., home control applications, meeting assistant, and city guide) in pervasive computing environments as long as their real-time requirements are not stringent.

5 Related Work

A lot of projects have been done in the area of context-aware computing in the past decade. Many have studied context-aware systems in feature-oriented approaches. AT&T Laboratories at Cambridge built a dense network of location sensors to maintain a location model shared between users and computing entities[12]. Microsoft's Easyliving focuses on a smart space that is aware of users' presence and adjusts environment settings to suit their needs[13]. HP's CoolTown provides physical entities (i.e., people, places and things) with 'Web presence', enabling users to navigate from the physical world to WWW by picking up links to Web resources using a variety of sensing technologies[14].There are other relevant projects in the area of smart spaces: Stanford's iRoom[15], MIT's Oxygen[16], CMU's Aura[17] to name a few. These projects have greatly contributed to the research of smart spaces by exploiting different features of pervasive computing.

A few projects specifically address the scalability and flexibility of context-aware applications by providing generic architectural supports. The seminal work of Context Toolkit[2] has provided an object-oriented architecture for the rapid prototyping of context-aware applications. Context Toolkit provides developers with a set of programming abstractions that help to separate context acquisition from actual context usage and to reuse sensing and processing functionality. Jason Hong et al. [18] have proposed an open infrastructure approach where underlying technologies are encapsulated into well-established services that can be used as a foundation for building applications. The European Smart-Its project have proposed a generic layered architecture for sensor-based computation of contexts, providing a programming abstraction that separate layers for raw sensor data, for features ex-

tracted from sensors ("cues"), and for abstract contexts derived from cues[19, 20]. Our work is similar to them in providing reusable architecture to ease application development. However, previous work does not provide adequate help on organizing contexts in a formal structured format. It is difficult for an independently-developed application to interpret contexts since the structure of contexts is not explicitly represented. Moreover, previous work can not provide a generic mechanism for the querying and reasoning of contexts. Simple matching mechanisms are employed to support the selective access of contexts. Developers often have to perform low-level programming to develop components that derive higher-level contexts.

Our work is different from, and perhaps outperforms previous work in many aspects. We use Semantic Web standards (i.e., RDF and OWL) to define context ontologies, providing the basis for building interoperable smart spaces where computing entities can easily exchange and interpret contexts based on explicit context representations. Enabled by Semantic Web technologies (i.e., knowledge base, query and inference), we have developed the context infrastructure with a generic mechanism for querying contexts using a declarative language and inferring higher-level contexts based on rules. Developers' work is greatly eased as they can realize expressive context querying and flexible context reasoning without programming.

6 Future Work

Semantic Space is our early efforts to incorporate Semantic Web technologies into pervasive computing environments. The use of Semantic Web technologies has helped developers to build smart spaces by providing support for explicit context representation, expressive context querying and flexible context reasoning. We envision several enhancements to our infrastructure. Currently the smart space uses local area network discovery protocol (i.e., UPnP) to dynamically locate and access Context Wrappers. In practical deployment, multiple smart spaces that belong to different users or parties with *private* contexts may share the same local network. Therefore there are many opportunities for the misuse of contexts, both from fraudulent context sources and misbehaving applications. To address privacy concerns, we will incorporate endpoint authentication into the context discovery process[21]. Each component in the infrastructure is associated with a URI and public-key certificate, which can be used to prove the component's identify to all other components. Smart spaces can specify both the Context Wrappers they trust and the Context Wrappers they have access to, thus the access of private contexts can be restricted to appropriate components.

Another enhancement is the support for uncertain contexts. Contexts provided by sources are not always precise, so we expect smart spaces to be capable to handle uncertainty. Based on the probabilistic extension to OWL ontology[22], we are working on extending context ontologies with the capability to capture uncertainty. Enhanced context ontologies will be able to support the probabilistic query with respect to quality of context, and the reasoning of uncertain contexts using various mechanisms(e.g., probabilistic logic, Bayesian networks, and fuzzy logic).

Reference

1. Schilit WN, "A Context-Aware Systems Architecture for Mobile Distributed Computing," PhD thesis, Columbia University, 1995.
2. A. K. Dey, "Providing Architectural Support for Building Context-Aware Applications," PhD thesis, Georgia Institute of Technology, 2000.
3. Tim Berners-Lee, et al., "The Semantic Web," *Scientific American*, May 2001.
4. Graham Klyne and Jeremy J. Carroll, editors, "Resource Description Framework (RDF): Concepts and Abstract Syntax," *W3C Recommendation*, 2004.
5. Deborah L. McGuinness and Frank van Harmelen, "OWL Web Ontology Language Overview," *W3C Recommendation*, 2004.

6. Karen Henricksen, et al., "Modeling Context Information in Pervasive Computing Systems," *Pervasive Computing and Communication (PerCom)*, 2002.
7. T. Gruber, "A Translation Approach to Portable Ontology Specifications," *Knowledge Acquisition*, 5(2): 199-220, 1993.
8. X. Wang, et al., "Ontology-Based Context Modeling and Reasoning using OWL," *Context Modeling and Reasoning Workshop* at *PerCom* 2004.
9. Y. Ding and D. Fensel, " Ontology Library Systems: The key for successful Ontology Reuse.," *International Semantic Web Working Symposium (SWWS)*, 2001.
10. Libby Miller, et al., "Three Implementations of SquishQL, a Simple RDF Query Language," *International Semantic Web Conference (ISWC)* 2002.
11. Jeremy J. Carroll, et al., "Jena: Implementing the Semantic Web Recommendations," Technical Report, HPL-2003-146, HP Laboratories Bristol.
12. A. Harter et al., "The Anatomy of a Context-Aware Application," *MobiCom* 1999, Seattle, WA.
13. B. Brumitt et al., "EasyLiving: Technologies for intelligent environments," *HUK* 2000, Bristol, UK.
14. T. Kindberg et al., "People, Places, Things: Web Presence for the Real World," *WMCSA* 2000.
15. Johanson, B., A. Fox, and T. Winograd, "The Interactive Workspaces Project: Experience with Ubiquitous Computing Rooms," *IEEE Pervasive Computing* 2002. 1(2): p. 67-74.
16. Dertouzos, M., "The Future of Computing," *Scientific American*, August 1999.
17. Garlan, D., et al., "Project Aura: Towards Distraction-Free Pervasive Computing," *IEEE Pervasive Computing*, April-June 2002.
18. Jason I. Hong and James A. Landay, "An Infrastructure Approach to Context-Aware Computing," *Human-Computer Interaction*, 2001, Vol. 16.
19. H. Gellersen, et al., "Multi-Sensor Context-Awareness in Mobile Devices and Smart Artifacts," *Mobile Networks and Applications (MONET)*, Oct 2002.
20. H. Gellersen, et al., "Physical Prototyping with Smart-Its," *IEEE Pervasive Computing*. Oct.-Dec. 2003.
21. Steven E. Czerwinski, et al., "An Architecture for a Secure Service Discovery Service," *International Conference on Mobile Computing and Networks (MobiCom)*, 1999.
22. Z. Ding and Y. Peng, "A Probabilistic Extension to Ontology Language OWL," 37th *Hawaii International Conference on System Sciences*, Hawaii, January 2004.