Semantic Space: An Infrastructure for Smart Spaces

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

Xiaohang Wang, Jin Song Dong, ChungYau Chin, and Sanka Ravipriya Hettiarachchi
National University of Singapore

Daqing Zhang
Institute for Infocomm Research, Singapore

Current pervasive computing research tries to merge the material and digital worlds by incorporating physical and computing entities into *smart spaces*. Homes, workplaces, classrooms, vehicles, and other spaces use embedded sensors, augmented appliances, stationary computers, and mobile handheld devices to gather information about users' locations, companions, and other aspects of their activities. Applications in such environments must be context aware so that they can adapt to rapidly changing conditions as users move about in their environments.[1,2]

The dynamic nature of smart spaces poses several challenges in developing context-aware applications. For example:

Rachel wishes to contact her friend Joey, so she instructs her mobile phone to arrange a call. Upon request, Joey's mobile phone checks the calendar and realizes he's currently attending a seminar. The phone determines on his behalf that he shouldn't be interrupted and schedules a call back when the seminar ends. Soon after the seminar, Professor Geller asks Joey 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 receiving the call. Based on contextual information (Where are you? Who are you with? What is the noise level? Is the door open or closed?) gathered in the smart space, the phone infers that Joey is in a conversation with his supervisor and decides to postpone the call until he's available. A few minutes later, when the conversation ends and Joey leaves the office, the phone finally reminds him of the missed call.

Building smart spaces relies on many different technologies, some of which we discuss in the "Related Work" sidebar. We focus here on three key issues:

- *Explicit representation*. Raw context data obtained from various sources comes in heterogeneous formats, and applications without prior knowledge of the context representation can't use the data. So, an interoperable smart space requires a way to explicitly represent context meanings (or semantics) so that independently developed applications can easily understand them.
- *Context querying*. A smart space maintains many contexts, and applications might need to selectively access a subset of them. The smart space should be able to answer expressive context queries—for example, who is in the room with Joey? When will the seminar he is attending or presenting end?
- *Context reasoning*. Higher-level contexts (What is the user doing? What is the activity in the room?) augment context-aware applications by providing summary descriptions about a user's state and surroundings. Although sensors can't recognize such contexts, they provide information that lets applications infer basic contextual information.

Published by the IEEE CS and IEEE ComSoc ■ 1536-1268/04/$20.00 © 2004 IEEE

Related Work

Many context-aware computing projects in the past decade have studied feature-oriented approaches to context-aware systems. AT&T Laboratories at Cambridge built a dense network of location sensors to maintain a location model shared between users and computing entities.[1] Microsoft's EasyLiving focuses on a smart space that is aware of users' presence and adjusts environment settings to suit their needs.[2] Hewlett Packard's CoolTown provides physical entities (people, places, and things) with "Web presence" and lets users navigate from the physical world to the Web by picking up links to Web resources using various sensing technologies.[3] Other relevant projects include Stanford University's iRoom,[4] the Massachusetts Institute of Technology's Oxygen,[5] and Carnegie Mellon University's Aura,[6] to name a few. These projects have greatly contributed to smart space research by exploiting different pervasive computing features.

A few projects specifically address the scalability and flexibility of context-aware applications by providing generic architectural supports. The seminal work of Context Toolkit[7] provides an object-oriented architecture for rapid prototyping of context-aware applications. Context Toolkit gives developers a set of programming abstractions that separate context acquisition from actual context usage and reuse sensing and processing functionality. Jason Hong and his colleagues[8] proposed an open-infrastructure approach that encapsulates underlying technologies into well-established services that can be used as a foundation for building applications. The European Smart-Its project proposed a generic layered architecture for sensor-based context computation, providing a programming abstraction that separates layers for raw sensor data, features extracted from sensors (cues), and abstract contexts derived from cues.[9,10]

Our work resembles these projects in providing a reusable architecture to ease application development. However, previous work doesn't provide adequate support for organizing contexts in a formal structured format. An independently developed application can't easily interpret contexts that have no explicitly represented structure. Moreover, previous work provides no generic mechanism for context querying and reasoning. Some models use only simple matching mechanisms to support selective context access, and developers must often perform low-level programming to develop components that derive higher-level contexts.

Our work differs from and perhaps outperforms previous work in several respects. Using the Semantic Web standards RDF and OWL to define context ontologies provides a foundation for building interoperable smart spaces where computing entities can easily exchange and interpret contexts based on explicit context representations. Using the Semantic Web technologies for knowledge base, query, and inference, we developed the context infrastructure with a generic mechanism for querying contexts using a declarative language and inferring higher-level contexts based on rules. This facilitates developers' work because they can realize expressive context querying and flexible context reasoning without programming.

REFERENCES

1. A. Harter et al., "The Anatomy of a Context-Aware Application," *Proc. 5th Ann. ACM/IEEE Int'l Conf. Mobile Computing and Networking* (MobiCom 99), ACM Press, 1999, pp. 59–68.
2. B. Brumitt et al., "EasyLiving: Technologies for Intelligent Environments," *Proc. 2nd Int'l Symp. Handheld and Ubiquitous Computing* (HUC 2000), LNCS 1927, Springer-Verlag, 2000, pp. 12–29.
3. T. Kindberg et al., "People, Places, Things: Web Presence for the Real World," *Proc. 3rd IEEE Workshop Mobile Computing Systems and Applications* (WMCSA 2000), IEEE CS Press, 2000, p. 19.
4. B. Johanson, A. Fox, and T. Winograd, "The Interactive Workspaces Project: Experience with Ubiquitous Computing Rooms," *IEEE Pervasive Computing*, vol. 1, no. 2, 2002, pp. 67–74.
5. M. Dertouzos, "The Future of Computing," *Scientific Am.*, Aug. 1999.
6. D. Garlan et al., "Project Aura: Towards Distraction-Free Pervasive Computing," *IEEE Pervasive Computing*, vol. 1, no. 2, 2002, pp. 22–31.
7. A.K. Dey, *Providing Architectural Support for Building Context-Aware Applications*, doctoral dissertation, Georgia Inst. Technology, 2000.
8. J.I. Hong and J.A. Landay, "An Infrastructure Approach to Context-Aware Computing," *Human-Computer Interaction*, vol. 16, nos. 2–4, 2001, p. 97.
9. H. Gellersen et al., "Multi-Sensor Context-Awareness in Mobile Devices and Smart Artifacts," *Mobile Networks and Applications (MONET)*, vol. 7, no.5, 2002, pp. 341–351.
10. H. Gellersen et al., "Physical Prototyping with Smart-Its," *IEEE Pervasive Computing*, vol. 3, no. 3, 2004, pp. 74–82.

To address these issues, we developed a context infrastructure called Semantic Space. We were inspired by the Semantic Web, which helps computers and people work better together by giving content well-defined meanings.[3] Using standards to represent machine-interpretable information (including RDF (resource description framework)[4] and OWL (Web Ontology Language)[5]), the Semantic Web offers a united approach to knowledge management and information processing. We explored OWL, RDF and their supporting tools by building a pervasive computing infrastructure with an ontology-based context model and a context infrastructure.

The context model

Context presentation is an important part of pervasive computing environ-

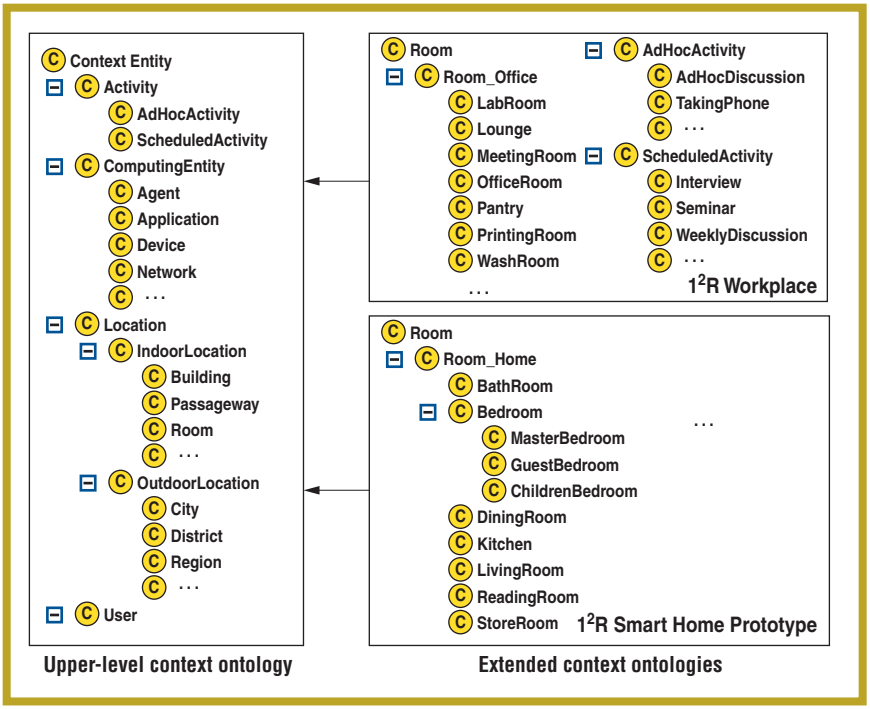


Figure 1. An upper-level context ontology and extended context ontologies.

ments. Because context-aware applications must adapt to changing situations, they need a detailed model of users' activities and surroundings that lets them share users' perceptions of the real world.[6]

An ontology approach to context modeling

Within the domain of knowledge representation, the term *ontology* refers to 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, OWL defines and instantiates Web information ontologies that let Web agents exchange and interpret information based on a common vocabulary. Using ontologies to model contexts in pervasive computing environments offers several advantages:

- By allowing pervasive computing entities to share a common understanding of context structure, OWL ontologies enable applications to interpret contexts based on their semantics.
- Ontologies' hierarchical structure lets developers reuse domain ontologies (for example, of people, devices, and activities) in describing contexts and build a practical context model without starting from scratch.
- Because contexts described in ontologies have explicit semantic representations, Semantic Web tools such as federated query, reasoning, and knowledge bases can support context interpretation. Incorporating these tools into smart spaces facilitates context management and interpretation.

Designing the context model

Smart spaces cover a range of environment types such as homes, offices, workplaces, classrooms, and vehicles. Rather than try to completely model all contexts in different kinds of smart spaces, we define an *upper-level context ontology* (ULCO) to provide a set of basic concepts common across different environments.[8] Among various contexts, we identify three classes of real-world objects (user, location, and computing entity) and one class of conceptual objects (activity) that characterize smart spaces. Linked together, these objects form the skeleton of a contextual environment. They also provide primary indices into other associated contexts. For example, given a location, we can acquire related contexts such as noise, weather, and the number of people inside if we model these objects as top-level classes in ULCO.

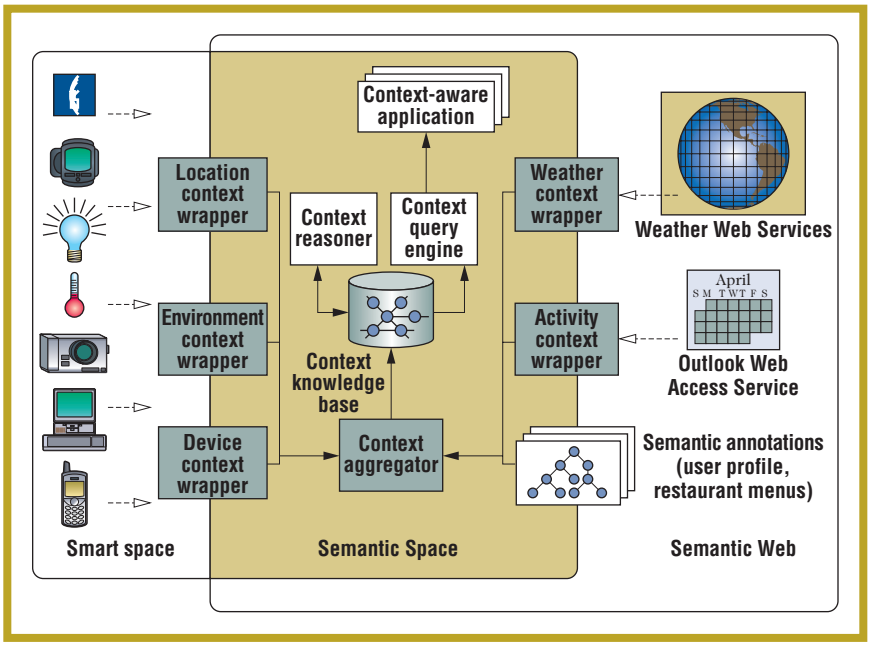
Knowledge reuse is one important advantage of ontologies.[9] We integrated consensus domain ontologies such as friend-of-a-friend (FOAF, http://xmlns.com/foaf/0.1), RCAL Calendar (www.daml.ri.cmu.edu/Cal), and FIPA Device Ontology (www.fipa.org/specs/fipa00091/XC00091D.html) into ULCO to model user, activity, and device contexts, respectively. These well-defined ontologies provide generic vocabularies that suit context ontologies' requirements—we need only add additional properties useful to smart spaces. For example, FOAF defines simple relationships between people (that is, `friendOf`), but we extend it to support richer properties such as `supervisorOf`, `studentOf`, and `colleagueOf`.

To let developers customize the context model for a particular smart space, we designed it to complement the classes defined in ULCO. A new application that needs additional classes can obtain them by inheritance from the ULCO classes, forming an *extended context ontology*, as Figure 1 shows. This lets developers easily build detailed context models for new smart spaces. Moreover, by providing shared terms and definitions, an ULCO supports better interoperability between extended context ontologies.

Marking up real-world contexts

Our model represents contexts as ontology instances and associated properties (*context markups*) that applications can easily interpret. Real-world contexts often originate from diverse sources, leading to dissimilar approaches to generating context markups. For example, in the smart phone scenario, some contexts (such as a person's name, relation-

Figure 2. The Semantic Space context infrastructure.



ships, and scheduled seminar time) have relatively slow change rates, and users often supply the information. Users also usually generate markups of these contexts. Our JavaScript application lets users create online profiles based on the ontology class User. The following example shows the context markup that describes RossGeller. (Throughout the article, we assume www.i2r.a-star.edu.sg/SemanticSpace# as the default base namespace.)

```
<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>
    <supervisorOfrdf:resource=
        "#JoeyTribbiani"/>
    <! —More properties not shown in this
        example—>
</User>
```

Hardware and software sources usually provide other contexts such as location, current time, noise level, or door status. Automated programs must mark up these contexts because they change frequently. Consider the RFID (radio frequency identification) indoor location system that tracks users' location by detecting the presence of body-worn tags. When Ross Geller enters Room 209, the RFID sensor detects his presence and composes the following context markup:

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

Each OWL instance has a unique URI, and context markups can link to external definitions through these URIs. For example, the URI www.i2r.a-star.edu.sg/SemanticSpace#RossGeller refers to the user we just defined, and the URI www.i2r.a-star.edu.sg/SemanticSpace#Room209 refers to a specific room defined elsewhere in the system.

The Semantic Space infrastructure

By representing contexts as easily interpreted semantic markups, the Semantic Space context infrastructure lets applications retrieve contexts using declarative queries and supports the inference of higher-level contexts from basic contexts. One such infrastructure is maintained in each smart space, which the physical space often bounds in a nonrestrictive way. For example, we can join two or more rooms to form a smart space or split a single room into multiple smart spaces.

The context infrastructure consists of several collaborating components: wrappers, an aggregator, a knowledge base, a query engine, and a reasoner (see Figure 2).

Context wrappers

Context wrappers obtain raw context information from various sources such as hardware sensors and software programs and transform them into context markups. Some context wrappers, including the location context wrapper, the environment context wrapper (which gathers environmental information such as temperature, noise, and light from embedded sensors), and the door status context wrapper (which reports the open or closed status of doors in each room), work with the hardware sensors deployed in our prototypical smart space. Software-based context wrappers include the activity context wrapper, which extracts schedule information from Outlook Web Access; the device context wrapper, which monitors the status of different networked devices (such as voice over IP or mobile phones); the application context wrapper, which monitors the status (idle, busy, closed) of applications such as JBuilder, Microsoft Word, and RealPlayer from their CPU usage; and the weather context wrapper, which periodically queries a Weather Web Service (www.xmethods.com) to gather local weather information.

All context wrappers are self-configuring components that support a unified interface for acquiring contexts from sensors and providing context markups to consumers (applications and the context aggregator). We implemented these wrappers as Universal Plug and Play (www.upnp.org) services that can dynamically join a smart space, obtain IP addresses, and multicast their presence

```
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)
```

Figure 3. Sample rules that infer a user's likely situation based on context, activity, location, and computing entity.

for others to discover. Context wrappers use the UPnP general event notification architecture (GENA) to publish context changes as events to which consumers can subscribe.

Context aggregator

This component discovers context wrappers and gathers context markups from them. We implemented the context aggregator as a UPnP control point that inherits the capability to discover context wrappers and subscribe to context events. Once a new context wrapper is attached to the smart space, the context aggregator will discover it, register it in the service directory, and obtain context markups from it. It asserts gathered context markups into the context knowledge base, which it updates whenever a context event occurs.

Context knowledge base

Residing in each smart space, CKBs provide persistent context knowledge storage. A CKB stores the extended context ontology for that particular space and the context markups that are given by users or gathered from context wrappers. The CKB links the context ontology and markups in a single semantic model and provides interfaces for the context query engine and context reasoner to manipulate correlated contexts.

Contexts in smart spaces display very high change rates, so the aggregator must regularly update the CKB with fresh contexts. The scope of contexts that the CKB manages also changes depending on the availability of wrappers. Developers can add a new wrapper to expand the scope of contexts in a smart space or remove an existing wrapper when the contexts it provides are no longer needed. The aggregator monitors the wrappers' availability and manages the scope of contexts in the CKB. When a context wrapper joins the smart space, the context aggregator adds the provided contexts to the CKB, and when the wrapper leaves, the aggregator deletes the contexts it supplied to avoid stale information.

Context query engine

The context query engine provides an abstract interface for applications to extract desired contexts from the CKB. To support expressive queries, we adopted the RDF Data Query Language[10] as the context query language. RDQL supports querying over semantic models based on triple (<subject, predicate, object>) patterns. This lets applications selectively access contexts using declarative statements. The following self-explanatory query statement asks, When will the ongoing seminar where Ross Geller is either 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>)
```

Context reasoner

This component infers abstract, higher-level contexts from basic sensed contexts. Semantic Space explicitly represents all contexts such that general-purpose reasoning engines can directly process them, making it easy for developers to realize application-specific inferences simply by defining heuristic rules.

Application-specific rules might generate conflicting results, but the reasoner doesn't assert inferred contexts into the CKB, thus avoiding conflict in a single model. When the application needs certain higher-level contexts, it submits a set of rules to the context reasoner, which applies them to infer higher-level con-

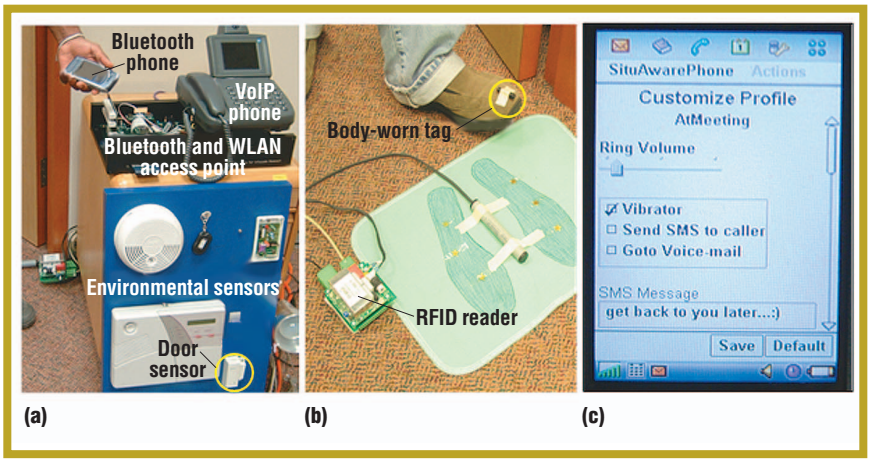


Figure 4. Building a prototype: (a) Networked sensors and devices; (b) the RFID indoor location system; and (c) a snapshot of SituAwarePhone, the GUI for configuring the response mode in each situation.

texts on the application's behalf, then returns newly inferred contexts without storing them in the CKB.

Our current system uses the Jena2 generic rule engine[11] to perform forward-chaining reasoning over the CKB. Developers can write rules for a particular application based on its needs. For example, the rules in Figure 3, related to the original scenario we presented, infer a user's likely situation based on context (such as identity and relationships with others), activity (such as type and time interval), location (such as location, office location, and others at the same location), and computing entity (such as status and ownership). The first rule examines whether a given person is currently engaged in a meeting on the basis of location and schedule—if he's in the meeting location and the current time (returned by currentDateTime()) is within the meeting's scheduled interval, he's likely to be at the meeting.

Implementation and evaluation

Using our workspace in the Institute for Infocomm Research (I2R) building as a test bed, we built a prototype of our context infrastructure. We defined the ULCO using OWL, based on which we developed an extended context ontology for our workplace. We implemented the CKB, context reasoner, and context query engine using the Jena2 Semantic Web Toolkit, and the discovery and event notification mechanism using Siemens UPnP SDK v1.01.

In addition to the general infrastructure, we provided context wrappers for sensing and markup of various contexts including location, schedule, temperature, noise, light, door status, device status, and application status. Figure 4a shows some of the networked sensors and devices we used to provide context information. Figure 4b shows the indoor location system we developed using TI-RFID Serial 2000.

Pervasive computing systems prove difficult to evaluate because they often stress new functionality and usability over pure performance. We believe our infrastructure's most important aspect is what it enables: representing contexts as semantic markups that applications can easily interpret, retrieving contexts using declarative queries, and supporting the inference of higher-level contexts from basic sensed contexts. Encapsulating these features into the API helps developers build applications that would otherwise be difficult to build.

We evaluated Semantic Space in two phases. First, we evaluated the infrastructure by building a real-world application using it. We then measured the resulting system's performance.

Application development

The widespread use of mobile phones raises social problems when, for example, phones ring during meetings or important conversations. Users often have to change phone settings according to their circumstances to avoid inappropriate usage. And frequent interactions with mobile phones impose significant user distractions. To solve this problem, we developed a context-aware application, SituAwarePhone (see Figure 4c), that adapts mobile phones to changing situations while minimizing user distractions.

As described earlier, when SituAwarePhone receives an incoming call, it first infers the user's situation using a set of rules, then automatically adapts its response mode (for example, adjust volume, set vibration, schedule a call back, send message, or forward to voice mailbox). SituAwarePhone also queries the smart space for various contexts that help in adaptation. For example, it asks for the end time of the seminar the owner is engaged in to schedule a callback, or it takes account of the caller's identity and identities of other people in a conversation to decide whether to let the call interrupt it.

In a smart space, the context infrastructure supports the entire process of gathering contexts from sources, managing contexts using the knowledge base, handling application queries, and reasoning about contexts based on rules. The infrastructure's simple client-side API lets applications access its functionalities while hiding the complexity of underlying context processing. Applications can use the following API methods to deal with contexts:

```
SemanticSpace(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
```

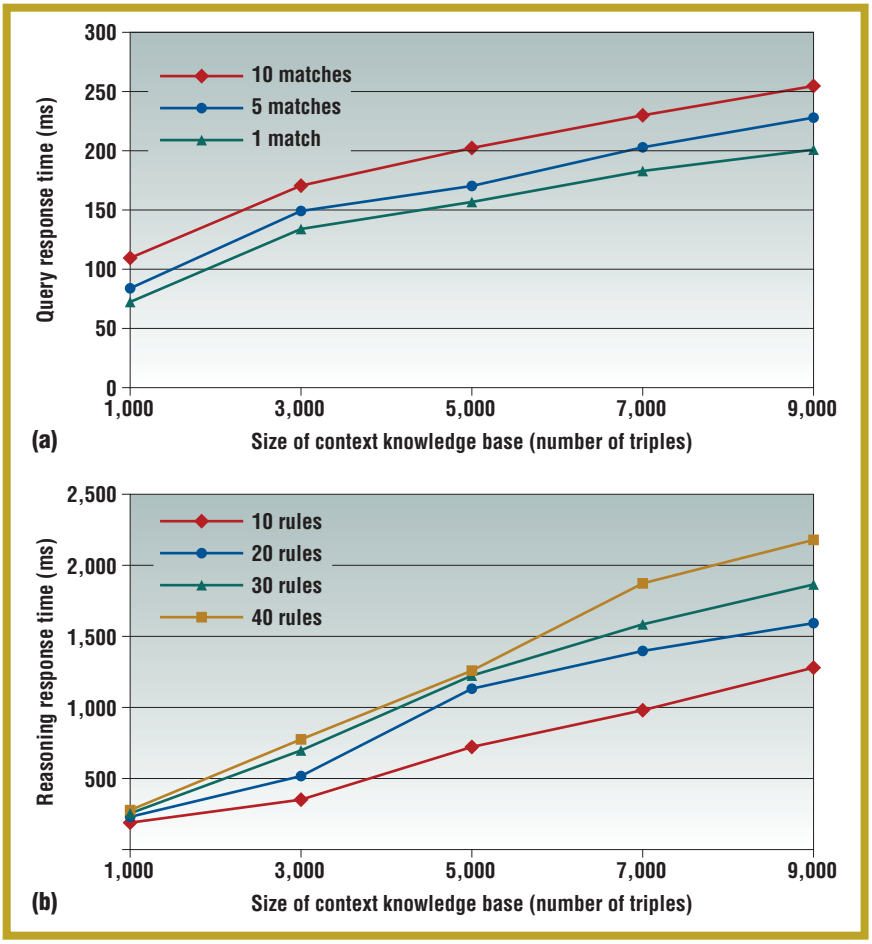


Figure 5. Query performance (a) and reasoning performance (b).

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

Using the client-side API, we built SituAwarePhone on top of the SonyEricsson P900 mobile phone (Figure 4c). The implementation amounts to approximately 800 lines of Java code, most of which deals with the MIDlet GUI and different response modes. Only about 20 lines of application-side code deal with contexts (to import libraries, issue queries, perform reasoning, parse returned models, and handle exceptions). This example shows how the context infrastructure can greatly ease the development of context-aware applications.

Performance

We evaluated system performance by measuring context querying and reasoning response times on a 2.4-GHz Pentium 4 workstation with 1.0 Gbyte of RAM running Redhat 9.0. We used five context data sets to test the system's scalability. Among these, the one with 3,000 triples (or 600 OWL classes and instances) is the real data set used in our prototypical smart space, while the other four are synthetic data sets with unique classes and instances.

We used the complex query statement described earlier to measure context-querying performance, varying the number of matched results from 1 to 10. Figure 5a shows the results—we expected the response time to be loosely proportional to the size of the context data set and the number of matched results, which the experiment corroborated. We then used four rule sets to test the context reasoner's performance, starting with SituAwarePhone's 10-rule set and using it to create others with increasing numbers of rules. The results (Figure 5b) show that rule-based reasoning is computationally intensive and the response time largely depends on the size of the data set and rule set applied.

As the smart space's scale increases, the reasoning response time will be human perceivable. However, our performance evaluation suggests that, in practice, rule-based context reasoning works for a useful set of pervasive computing applications. Taking SituAwarePhone as an example, before the mobile phone responses to an incoming call, it must reason about the callee's situation. Context reasoning causes a perceivable delay (about one second), which sometimes matters to users. For example, the callee might wish to answer the call from his supervisor as soon as possible, and the caller always expects a quick response. But most of our invited evaluators, both as callers and callees, found SituAwarePhone's response time acceptable. We also believe the context infrastructure can support many applications (such as home control, meeting assistant, and city guide) in pervasive computing environments as long as their real-time requirements are not stringent.

Semantic Space represents our early efforts to incorporate Semantic Web technologies into pervasive computing environments. Semantic Web technologies have helped developers 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 the local-area-network discovery protocol UPnP to dynamically locate and access context wrappers. In practical deployment, multiple smart spaces that belong to different users or parties with private contexts might share

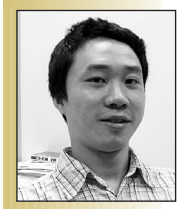
the same local network. This presents many opportunities for context misuse from both fraudulent context sources and misbehaving applications. To address privacy concerns, we'll incorporate endpoint authentication into the context discovery process.[12] Each infrastructure component is associated with a URI and public-key certificate, which can be used to prove its identity to all other components. Smart spaces can specify both the context wrappers they trust and those they have access to, thus restricting access of private contexts to appropriate components.

We also plan to provide support for uncertain contexts, because source-based contexts aren't always precise. Using the OWL ontology's probabilistic extension,[13] we're working on extending context ontologies to capture uncertainty and give smart spaces the ability to handle it. Enhanced context ontologies will support probabilistic querying with respect to context quality and the reasoning of uncertain contexts using such mechanisms as probabilistic logic, Bayesian networks, and fuzzy logic. P

REFERENCES

1. W.N. Schilit, *A Context-Aware Systems Architecture for Mobile Distributed Computing*, doctoral dissertation, Columbia Univ., 1995.
2. A.K. Dey, *Providing Architectural Support for Building Context-Aware Applications*, doctoral dissertation, Georgia Inst. Technology, 2000.
3. T. Berners-Lee et al., "The Semantic Web," *Scientific Am.*, May 2001.
4. G. Klyne and J.J. Carroll, eds., "Resource Description Framework (RDF): Concepts and Abstract Syntax," *W3C Recommendation*, 2004.
5. D.L. McGuinness and F. van Harmelen, "OWL Web Ontology Language Overview," *W3C Recommendation*, 2004.
6. K. Henricksen, J. Indulska, and A. Rakotonirainy, "Modeling Context Information in Pervasive Computing Systems," *Proc. 1st Int'l Conf. Pervasive Computing* (Pervasive 2002), LNCS 2414, Springer-Verlag, 2002, pp. 167–180.
7. T. Gruber, "A Translation Approach to Portable Ontology Specifications," *Knowledge Acquisition*, vol. 5, no. 2, 1993, pp. 199–220.
8. X. Wang et al., "Ontology-Based Context Modeling and Reasoning Using OWL," *Proc. 2nd IEEE Conf. Pervasive Computing and Communications (PerCom 2004) Workshop on Context Modeling and Reasoning*, IEEE CS Press, 2004, pp. 18–22.
9. Y. Ding and D. Fensel, "Ontology Library Systems: The Key for Successful Ontology Reuse," *Proc. 1st Semantic Web Working Symp.* (SWWS 01), Stanford Univ. Press, 2001, pp. 93–112.
10. L. Miller, A. Seaborne, and A. Reggiori, "Three Implementations of SquishQL, a Simple RDF Query Language," *Proc. 1st Int'l Semantic Web Conf.* (ISWC 2002), LNCS 2342, Springer-Verlag, 2002, pp. 423–435.
11. J.J. Carroll et al., *Jena: Implementing the Semantic Web Recommendations*, tech. report HPL-2003-146, Hewlett Packard Laboratories Bristol, 2003.
12. S.E. Czerwinski et al., "An Architecture for a Secure Service Discovery Service," *Proc. 5th Ann. ACM/IEEE Int'l Conf. Mobile Computing and Networking* (MobiCom 99), ACM Press, 1999, pp. 24–35.
13. Z. Ding and Y. Peng, "A Probabilistic Extension to Ontology Language OWL," *Proc. 37th Hawaii Int'l Conf. System Sciences* (HICSS 04), IEEE CS Press, 2004.

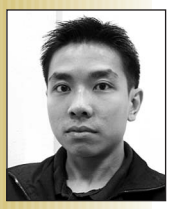
the AUTHORS



Xiaohang Wang is a graduate student at the National University of Singapore. His research interests include pervasive and context-aware computing, the Semantic Web, and home networking. He received his BS in computer science from Huazhong University of Science and Technology, China. Contact him at the Institute for Infocomm Research, MailBox #B097, 21 Heng Mui Keng Terrace, Singapore 119613; xwang@i2r.a-star.edu.sg.

Daqing Zhang is head of the Context-Aware Systems Department at the Institute for Infocomm Research, where he leads research in connected-home and context-aware systems. His research interests include pervasive computing, service-oriented computing, context-aware systems, and home networking. He received his PhD from University of Rome La Sapienza and University of L'Aquila, Italy. Contact him at the Institute for Infocomm Research, 21 Heng Mui Keng Terrace, Singapore 119613; daqing@i2r.a-star.edu.sg.

Jin Song Dong is a faculty member at the National University of Singapore. His research interests include formal methods, Web-based software design, and programming-language semantics. He received his PhD in computing from the University of Queensland. Contact him at 10 Kent Ridge Crescent, Singapore 119260; dongjs@comp.nus.edu.sg.



ChungYau Chin is a graduate student in the Department of Electrical and Computer Engineering at the National University of Singapore. His research interests include pervasive computing, context-aware systems, service discovery, the Semantic Web, and networking. He received his BEng in computer engineering from the National University of Singapore. Contact him at 4 Amber Rd., Amber Tower #03-04, Singapore 439852; chungyau.chin@nus.edu.sg.

Sanka Ravipriya Hettiarachchi is an undergraduate student in the Department of Electrical and Computer Engineering, National University of Singapore. His research interests include context-aware systems mobile computing, and smart-phone application development. Contact him at #14-249, Block 508, West Coast Dr., Singapore 120580; eng12004@nus.edu.sg.