International Journal of Web Services Research , Vol.X, No.X, 200X

Ubiquitous Provision of Context-Aware Web Services

Stephen J.H. Yang
National Central University, Taiwan
jhyang@csie.ncu.edu.tw

Jia Zhang
Northern Illinois University
jiazhang@cs.niu.edu

Irene Y.L. Chen
Ching Yun University, Taiwan
irene@cyu.edu.tw

ABSTRACT:

Context-aware Web services refers to an adaptive process of delivering contextually matched Web services to meet service requesters' needs at the moment. We define the term "context" from two perspectives: one from service requesters, and the other from Web services. From the former perspective, context is defined as the surrounding environment affecting requesters' services discovery and access, such as requesters' preferences, locations, activities, and accessible network and devices. From the latter perspective, context is defined as the surrounding environment affecting Web services delivery and execution, such as networks and protocols for service binding, devices and platforms required for service execution, and so on. This paper presents an ontology-based context model that enables formal description and acquisition of contextual information pertaining to service requesters and services. The context model is supported by context query and phrased acquisition techniques. We also report two context-aware Web services built on top of our context model to demonstrate how our context model can be used to facilitate Web services discovery and Web content adaptation. Implementation details of the context elicitation system and the evaluation results of context-aware services provision are also reported.

KEY WORDS:
Context-aware, handheld devices, OWL-S, service oriented architecture, ubiquitous, Web services

INTRODUCTION

When a Web service is developed, it typically requires its most suitable context (e.g., platforms and devices) for the best execution performance. For example, a Web service may be developed oriented to desktop browsers instead of Personal Digital Assistant (PDA) browsers. Meanwhile, as people are constantly on the move in nowadays heterogeneous working environment, resources (e.g., computational devices and communication network coverage) are more frequently prone to change due to physical location changes. Therefore, in the process of a service consumption, a service may need to smoothly adjust its content delivery adapting to the ever-changing

environment (i.e., context), especially when a gross mismatch between resource requirements and supplies occurs (Satyanarayanan, 2004). For example, if a service requester drives into a dark area, it will be good for the corresponding service to switch to audio content delivery. Providing context-aware Web services thus refers to an adaptive process of delivering contextually matched Web services to service requesters. In other words, it aims to provide personalized and adaptive services based on service requesters' varying characteristics and situated environments. We envision that providing context-aware Web services is the first step toward ubiquitous Web services by finding right services in the right place at the right time.

In our research, context not only emphasizes on people's mobility and physical location, but also on people's abstract situations (e.g., whether they are in a meeting). In this paper, the two terms "*users' situated environment*" and "*context*" are used interchangeably, both referring to surrounding information, from either service requesters or services, which may impact service execution including computational devices, communication network, lighting, noise level, location, activity, and time (Schilit, Adams, and Want, 1994; Dey & Abowd, 1999). We summarize the characteristics of context-aware Web services and their requirements in the following eight aspects: mobility, location awareness, interoperability, seamlessness, situation awareness, social awareness, adaptability, and pervasiveness.

(1) Mobility: The continuousness of computing capability while moving from one position to another. Requirements include mobile computing on portable devices with embedded software.
(2) Location awareness: The capability of detecting and identifying the locations of persons and devices. Requirements include outdoor positioning and indoor positioning.
(3) Interoperability: The capability of interoperable operation between various standards of resource exchange and services composition and integration. Requirements include standards of content, services, and communication protocols.
(4) Seamlessness: The capability of providing an everlasting service session under any connection with any device. Requirements include state transition of network roaming and service migration.
(5) Situation awareness: The capability of detecting and identifying person-situated scenarios. Requirements include knowing what a person is doing with whom at what time and where.
(6) Social awareness: The capability of knowing who are socially related? What do they know? And what are they doing at one moment? Requirements include knowing social partners' knowledge competence and social familiarity.
(7) Adaptability: The capability of dynamically adjusting services/contents depending on users' needs. Requirements include knowing people's accessibility and preferences.
(8) Pervasiveness: The capability of providing intuitive and transparent way of service/content access. Requirements include predicting what users want before their explicit expressions.

The characteristics of context-aware Web services pose significant challenges. Among others, such characteristics and constraints should be formalized with requirements specification, so that they can be precisely defined and captured in order to satisfy the demands of a service requestor in a ubiquitous environment.

For better explaining why contextual information is necessary for proper services provision and better explaining our solution, let us consider an example scenario that will be used throughout this paper. Steve is a manager, who just finished a product presentation in a trade fair and is now searching for a "Web business meeting" service to remotely discuss some timely issues (e.g., find

socially related partners). The "availability" of the service must be 99% or above (i.e., to meet location-aware QoS requirement). During the Web meeting, Steve needs to communicate with his colleagues using various devices to exchange multimedia-based information (i.e., provide device-independent content adaptation). In thirty minutes, Steve needs to drive back to his office for a pre-scheduled face-to-face meeting with another customer. He wants to continue the "Web business meeting" seamlessly with his colleagues while he is driving (i.e., provide situation-aware seamless interoperability). Hence, Steve needs to automatically switch to "PDA-based Web business meeting."

Based on Steve's requirements, a published Web service entitled "Web business meeting" may be a qualified candidate, which can support both PCs and PDAs via wireless LAN and General Packet Radio Service (GPRS), respectively, with a guarantee of 99% even when users are physically moving. However, it should be noted that many other contextual information could also affect how well Steve is served. For example, who else attend the conference? Who are Steve's colleagues and what are they doing during the meeting? What if Steve's colleagues are not available at the time? Could Steve find some other information or people with proper expertise? What kind of network channels and devices will Steve use to connect to the Web? How will various situations (e.g., driving) affect Steve's device usage, network access and services delivery? All of these kinds of contextual information need to be considered in order to provide better services provision according to Steve's requirements and conditions at the moment.

The main contribution of this paper is the development of a context model and two context-aware Web services built as proofs-of-concepts. We have developed a context model to formally describe and acquire contextual information pertaining to both service requesters and services. Based on the model, we have built two context-aware Web services for demonstrating how to find right services and right information (i.e., content presentation) using context specifications and requirements. In this paper, we refer the so-called "right" as "context-aware." The terms "information" and "content" are also used interchangeably.

The remainder of this paper is organized as follows. Related research regarding Web services and context-aware computing is first presented. Then we introduce our context model and the entailed context acquisition technique. Afterwards, we present two context-aware Web services and present experiments and discussions. Finally, we conclude the paper.

RELATED RESEARCH

The emerged Web services technology provides a flexible and efficient approach to reuse existing Web-based applications and services. Web service technology concentrates not only on interoperability but also on how to describe, publish, locate and invoke Web services. A number of standards and specifications created from industry and academia have contributed to the development of Web services, such as WSDL (http://www.w3.org/TR/WSDL) and UDDI (http://www.uddi.org/). Service providers can describe a service in WSDL to specify what it can do and how to invoke it. A UDDI registry creates a standard interoperable platform that enables businesses and applications to quickly, easily, and dynamically publish and locate Web services over the Internet. However, the above Web service infrastructure can only behave well at the syntactical level while lacking semantic information to support inferential capability. This motivates Web services a step further to semantic Web (Berners-Lee, Hendler, & Lassila, 2001). Semantic Web description languages, such as DAML+OIL (http:// www.daml.org/) and Web Ontology Language (OWL, http://www.w3.org/TR/owl-features/), provide predicate-like and description logic-based annotation of Web services to encourage inferential procedures on

annotated Web services. The combination of semantic Web's expressive power and the Web service infrastructure results in the concept of semantic Web services. As the first step toward semantic Web services, OWL-S (http://www.daml.org/services/owl-s/1.1/overview/) is a specific application for Web services by applying OWL on a service's capability representation. It supports Web service providers with a core set of markup language constructs for describing properties and capabilities of their Web services in an unambiguous, machine interpretable way. We choose OWL-S as the implementation language of our context-aware Web services.

Service oriented architecture (SOA) is a conceptual model that inter-relates services through well-defined interfaces and contracts between the services. An interface is defined in a neutral manner independent of hardware platforms, operating systems, and programming languages in which the services are implemented. This allows services to be built on a variety of environments to interact with each other in a uniform and universal fashion. Three fundamental roles exist in SOA: service providers, service requesters, and service matchmakers. The service providers publish services to the matchmakers; the service requesters send requests to the matchmakers on an on-demand basis; the matchmakers perform service matching to find available services and bind the services from the providers to the requesters. While Web services are moving toward semantic Web, the matchmakers in SOA also favor semantic-based service matchmaking.

LARKS is a semantic matchmaker that performs both syntactic and semantic matching (Sycara, Klusch, & Lu, 2002). It can identify different degrees of partial matching based on five filters: context matching, profile comparison, similarity matching, signature matching, and constraint matching. Requesters can select any desired combination of aforementioned filters based on different concerns, such as efficiency and computation cost. DAML-S/UDDI is another semantic matchmaker that expands UDDI to provide semantic matching (Sycara et.al., 2003). It can perform inferences based on sub-assumption hierarchy leading to the recognition of semantic matches regardless of their syntactical differences. A flexible matching strategy is adopted based on inputs and outputs to identify similarities between requests and service's advertisements. However, although DAML-S/UDDI matchmaker expands functionality of UDDI registry to enable capability match, it does not take requesters' contextual information into account. Requesters may receive some invalid services, and they need to manually verify all matched services. In contrast, our context-aware SOA can provide automatic and better contextually matched services to meet requesters' context changes in addition to services' contextual requirements.

Context can be interpreted differently from various perspectives. In this research, we will address context from the aspect of mobile and pervasive computing. Schilit, Adams, and Want (1994) point out that one of the major challenges of mobile computing is how to exploit the changing environment with a new class of applications that are aware of context. They believe context encompasses more than just people's location; instead, context should also include environment's lighting, noise level, network communication, and even people's social situation. They also believe that context-aware applications should be able to adapt to changing environment according to the location of use, the collection of nearby people, hosts, and accessible devices, as well as changes to such things over time.

Dey & Abowd (1999) define context as "*any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and application themselves*." They point out that certain types of context (e.g., identity, location, activity, and time) are, in practice, more important than others. Given a person's identity, on can acquire many pieces of related information, such as the person's contact information and preferences. Given a person's

location information, one can determine what other resources or people are nearby and what activities are occurring close to the person. Given a timestamp or a period of time, one can find out what events are taking place or what activities have occurred at the same time. Based on the four primary context types, Dey and Abowd (1999) further define a context-aware system as "*if it uses context to provide relevant information and/or services to the user, where relevance depends on the user's task.*" They also refer context-aware applications as applications that automatically provide information and take actions according to the user's present context as detected by sensors. In addition, context-aware applications look at *who*'s, *where*'s, *when*'s and *what*'s (i.e., what a person is doing), and use this information to determine *why* the situation is occurring.

In addition to the aforementioned definitions, there are many comprehensive survey articles and special issues on journals covering context and context-aware applications. Readers can refer to the articles such as Dey and Abowd's (1999) "Toward a Better Understanding of Context and Context-awareness," Korkea-aho's (2000) "Context-Aware Applications Survey," and Chen and Kotz's (2000) "A Survey of Context-Aware Mobile Computing Research". In addition, two special issues on journals can be found on "Context-aware computing": *IEEE Pervasive Computing* (Abowd, et. al., 2002) and *Human-Computer Interaction* (Moran and Dourish, 2001).

There are also many research efforts on the development of toolkits supporting context-aware services provision, including HP's Cooltown project (http://www.hpl.hp.com/archive/cooltown/), Dey's Context Toolkit (http://www.cs.cmu.edu/~anind/context.html), Mostefaoui et. al.'s (2003) CB-SeC framework, and Roman et al.'s (2002) Gaia middleware. These toolkits either provide functionalities to help service requesters find services based on their contexts, or enable content adaptations according to requester's contextual information.

Mostefaoui and Hirsbrunner (2004) propose a formal definition of service contexts to model a service's contextual information. Lemlouma and Layaida (2001) propose a framework to assert metadata information of Web contents. They both use Composite Capabilities/Preferences Profiles (CC/PP) as interoperable context representation to enable communication and negotiation of device capabilities and user preferences in terms of a services invocation. Zhang et al. (2005) further propose extensions to CC/PP to enable transformation descriptions between various receiving devices. Besides, several OWL-based context models are presented (Broens et. al., 2004; Khedr and Karmouch, 2004; Khedr, 2005) to provide high-quality results of services discoveries beyond the expressive limitations of CC/PP. These researchers all utilize ontology to describe contextual information including location, time, device, preference, and network.

However, the existing context definitions either focus on service requesters or on services alone. We argue that effective and efficient context-aware service provision needs to consider from both sides. In contrast to aforementioned related work, our approach stands out from three aspects: First, we formalize an ontology-based integrated context model considering both service requesters and services. Second, we provide a rule-based context elicitation technique with tailored contextual information collection. Third, we employ semantic matchmaking to enhance the recall and precision of context-aware service provision.

CONTEXT MODEL

In this section, we present our context model that is developed to formally define context description pertaining to service requesters and services. A context acquisition mechanism is designed for collecting contextual information at run time.

The underlying foundation of our context model is to treat a context-aware Web service as a parameterized abstract machine. An abstract machine is characterized by its static (i.e., state variables) and dynamic (i.e., state functions) characteristics. Contextual information is represented as functions that may change the state of the service. All functions are equipped with formally defined pre-conditions, post-conditions, and invariants based on the ontologies defined in our context model. The abstract machine structure is defined as follows:

```
MACHINE M(X,x)
ONTOLOGIES O
DEFAULTS D
SETS S; T={a,b}
PROPERTIES P
VARIABLES V
INVARIANT I
ASSERTIONS J
INITIALIZATION B
REQUIREMENTS
u1 <- O1(w1) = PRE Q1 THEN V1 END
…
un <- On(wn) = PRE Qn THEN Vn END
END
```

The abstract machine has free dimensions *X* (set) and *x* (scalar). *ONTOLOGIES* describes the semantic meanings of the machine parameters. *SETS* contains finite or named sets that the machine can use. *DEFAULTS* describes default values. *PROPERTIES* takes form of conjoined predicates specifying invariants involving defaults and sets. *VARIABLES* lists state variables, and *INVARIANT* describes static properties of the machine that must be maintained with contextual settings. *ASSERTIONS* is deducible from PROPERTIES and INVARIANT, and exists purely to ease the proving of machine correctness. *INITIALIZATION* initializes state variables. *REQUIREMENTS* lists possible requirements of the abstract machine, with pre-conditions *PRE* and post-conditions *THEN*.

Context Description

We conceive context awareness as an interactive model between service requesters and services; thus, context description is addressed from both parties. We have developed two types of context ontology for describing the circumstances of requesters and services, respectively: requester ontology and service ontology. The separation of concern allows higher flexibility and extensibility.

The major difference between the requester ontology and service ontology is their profiles. As follows, the requester ontology contains requester profiles such as personnel profile, accessibility and preferences, calendar profile, social profile, and location profiles.

```
Requester_ontology =
    {Profiles, Preferences, QoWS, Environment, Devices}
    Profiles = {Personnel, Location, Calendar, Social}
    Personnel_profile = {name, role, id, email, accessibility}
```

```
Location_profile = {office, building, home}
Calendar_profile = {owner, event , time, attendee*, location}
Social_profile = {owner, partner+}
QoWS = {Functional requirements, non-functional requirements}
Environment = {Network_channel, Situation}
Network_channel = {wired, wireless}
Situation = {normal, meeting, walking, driving}
Devices = {hardware, software}
```

On the other hand, the service ontology contains service profile such as input, output, pre-condition, and effect of service execution as follows.

```
Service_ontology = {Profile, QoWS, Environment, Devices}
Service Profile = {name, id, description, input, output,
   pre-condition, effect}
QoWS = {Functional requirement, non-functional requirement}
Environment = {Network channel, Situation}
Network channel = {wired, wireless}
Situation = {normal, meeting, walking, driving}
Devices = {hardware, software}
```

In addition to profiles, both requester ontology and service ontology contain a cluster of specifications reflecting surrounding context including Quality of Web Services (QoWS) specifications, environment specifications, and device capability specifications. QoWS specifications contain both functional and non-functional constraints. Functional QoWS constraints can be described using network bandwidth and response time; non-functional QoWS constraints can be described in reliability, availability, and cost. Environment specifications contain network channel constraints and situated location constraints. Network channel constraints can be used to describe the types of channels (e.g., wired or wireless); situated location constraints can be used to describe requester situated environment (e.g., in a meeting, reading, walking, or driving). Device capability specifications contain a device's hardware and software constraints. Various devices (e.g., PDAs and mobile phones) are associated with different hardware and software constraints. Hardware constraints can be used to describe the hardware requirements of a device such as platform, CPU speed, memory size, screen size, and resolution. Software constraints can be used to describe the software requirements of a device such as operating system, browser, playable media type, and resolution.

Context Acquisition

With our ontology models, both service requesters and services could formally define their contextual information accordingly. We define *context acquisition* as a process of obtaining the values of the defined properties using the requester ontology and service ontology. We separate the context acquisition function from the context aware services, which decoupling enables the reuse of existing context acquisition functions for various services.

Context acquisition can be conducted in three incremental ways: form-filling, context detection, and context extraction. In the *form-filling* approach, contextual information is acquired directly from requesters' inputs. In the *context detection* approach, various sensing, recording, and positioning systems (e.g., GPS, RFID, and sensor networks) are utilized for location detection. In the *context extraction* approach, contextual information is derived from requester ontology and

service ontology. The first approach form-filled can be used to construct personnel profiles, preferences, calendar profiles, and social profiles; the term form-filled explains for itself. We thus will concentrate on the other two approaches: context detection and context extraction.

Context Detection

Context detection aims to detect and analyze contextual information such as location, environment, and device profiles during the run time. We have designed a context detection environment that facilitates the process from both service side and service requester side. On the service requester side, smart devices and sensor networks are used to sense and react to the requesters' surrounding environment, as we reported in [14]. On the service side, we have designed a Web service portal to accept service requests featuring a recording capability [14]. Whenever a requester logs in, our portal catches the request, analyzes what kind of devices the requester is using to build the device profile, and detects what kind of network channel the requester is using to connect to the Internet to build the environment profile. Some situations, such as whether the requester is in a meeting or is driving, remain unknown at this stage. We defer this analysis to the context extraction phase. Besides detecting the request, our portal also records and keeps the history of service requests associated with every requester who registers in the portal. Based on the historical information, we can further conduct analysis about the requesting behaviours and requesting patterns that are important references for building requesters' preferences.

Our environment is equipped with a set of location detection Web services that can match all possible location tracking functionalities currently available for requesters' devices, and filter them based on requesters' actual contexts. For example, if a requester is outside of a building, then a GPS location detection service will be invoked to return her location in terms of building name or number. If the requester is inside of a building, then an indoor tracking service provided by RFID or sensor network will be invoked to return her location in terms of room number. Once the location is positioned, our location detection services can further decide whether to disclose the location based on requester's privacy preference. One thing worth noting is that in our environment, we consider privacy preference in a dynamic manner and can be adjusted based on location and temporal constraints. For example, if a requester is in her office during her office hours, she is willing to disclose her room number to her students; if she is out of town in a trip, she may only disclose her position to her colleagues and family members.

Context Extraction

If the context detection services cannot detect current context explicitly, requesters' profiles will be taken into consideration. We define *context extraction* as a process to derive contextual information from requesters' preferences and profiles. Comprehensive contextual information can be extracted combining static and dynamic approaches. The static context extraction elicits a requester's default context from her predefined preferences and personnel profiles. The dynamic context extraction deduces a requester's actual context from her calendar profiles and social profiles.

In the static approach, except for those required properties for which requesters must specify values (e.g., name, id, role, and email), other properties defined in personnel profiles and preferences may have predefined default values. As a result, our system fills in the default values for the requesters if they do not explicitly specify the property values. We refer this process as *context wrapping*, which will be addressed in more detail in the next section.

In the dynamic approach, we analyze a requester's calendar and social profiles to deduce her actual contextual information. Using our example earlier, by checking Steve's calendar profiles and social profiles, we can find out at 10:15am on Wednesday, he is in a meeting with his team members.

As shown below, we formally define a requester's calendar profile with properties that indicate the owner of the calendar, the privacy of using the calendar, the event title and description, the begin time and end time of the event, as well as the attendee and location of the event. Privacy is a property containing policy rules to determine whether this calendar is accessible for public or for private reference only.

```
Calendar_profile = {owner, event, time, attendee*, location}
   owner = {name, id, privacy}
   event = {title, description}
   time = {begin(yyyy:mm:dd;hh:mm), end(yyyy:mm:dd;hh:mm)}
   attendee = {name, contact_info}
   location = {place, contact_info}
```

A social profile is used to find the most related business partners when a requester does not explicitly specify with whom she is working. Social profiles are also useful when a requester makes her calendar profiles private but such information is needed to locate her contexts. This goal can be fulfilled by querying every partner's calendar profile to find the events associated with the target requester. A requester can have various types of business partners such as an individual or a group of team workers. The formal definition of social profile is shown as follows.

```
Social_profile = {owner, partner+}
  owner = {name, id, privacy}
  partner = {type, name, context_info},
        type = {individual | working_team | enterprise | community}
```

CONTEXT-AWARE WEB SERVICES

Based on the definitions of context model and the contextual information entailed from the context acquisition, we built two context-aware applications for demonstrating how to find right services, right partners (collaborators), and right information (content presentation) using our proposed approach. The two applications are: (1) an agent-based context-aware Web services provision environment, and (2) a context-aware Web content adaptation engine supporting mobile computing.

Context-aware Web Services Provision

Based on the contextual information collected by the context elicitation system, we present our context-aware service oriented architecture (CA-SOA) for providing Web services request, publication, and discovery. As shown in Figure 1, CA-SOA consists of three major components - an agent platform, a service repository, a context-aware matchmaker.

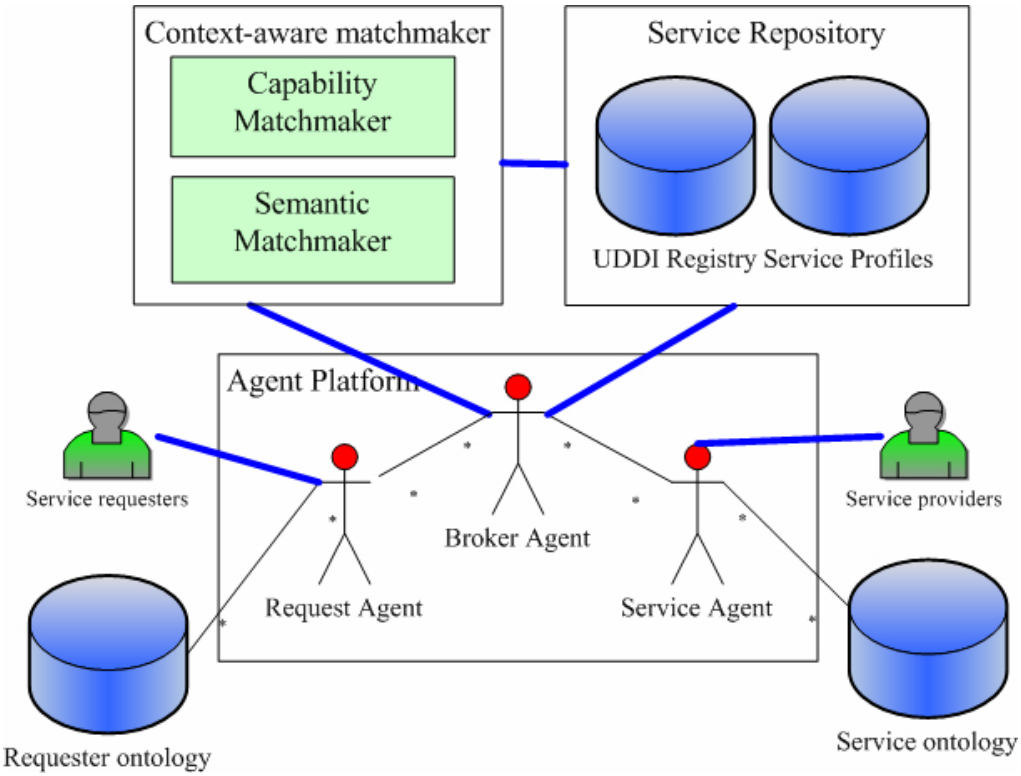

Figure 1. Context-aware SOA

To enhance context-aware services provision, we have implemented three categories of agents in the agent platform - service agents, broker agents, and request agents. Service agents are designed to help service providers formally describe and wrap Web services with contextual descriptions obtained from the service ontology. Request agents are designed to help service requesters formally describe and wrap their service requests with requesters' contextual information obtained from the requester ontology. On one hand, broker agents take services' publishing requests from service agents and save service descriptions and contextual descriptions into service profiles; on the other hand, broker agents take requests from request agents and initiate context-aware matchmaking, which consists of two phases- capability matchmaking and semantic matchmaking.

As shown in Figure 1, our service repository is designed to encompass a general UDDI Registry associated with service profiles. If the required services cannot be found by a capability matchmaking process in the UDDI Registry, the semantic matchmaker will be invoked. The semantic matchmaker will decompose service requests into a set of sub-requests, and then schedule an integrated composite service based on the decomposed requests.

Based on the CA-SOA, we present how to provide context-aware service request, publication, and discovery in the following sub-sections.

Requests' Contextual Description and Wrapping

Service requests are generally specified with keywords or descriptions regarding inputs, outputs, pre-conditions, and effects of the requests. However, it will cause tremendous overhead if requesters need to manually input much contextual information. Thus, we utilize request agents to

automatically wrap requests' contextual information. Follows is a stepwise procedure to describe and wrap requests' contextual information and transform it into a request package.

1. A request agent provides a request parser as an interface (shown in Figure 2) for guiding requesters to input their request descriptions, including service name, service key, service description, provider name, provider key, TModel name, TModel description, and context information.
2. The request parser transforms request descriptions to OWL-S.
3. The request agent instantiates the requester ontology to generate instances of requesters' static context profiles.
4. The request agent provides a context wrapping service for obtaining requesters' static contextual information by tagging the attributes defined in requester's instances of static context profiles.
5. The request agent enacts a context elicitation service for obtaining requesters' dynamic contextual information.
6. The request agent wraps and packages the request along with requesters' contextual information.
7. The request agent sends the request package to a broker agent.

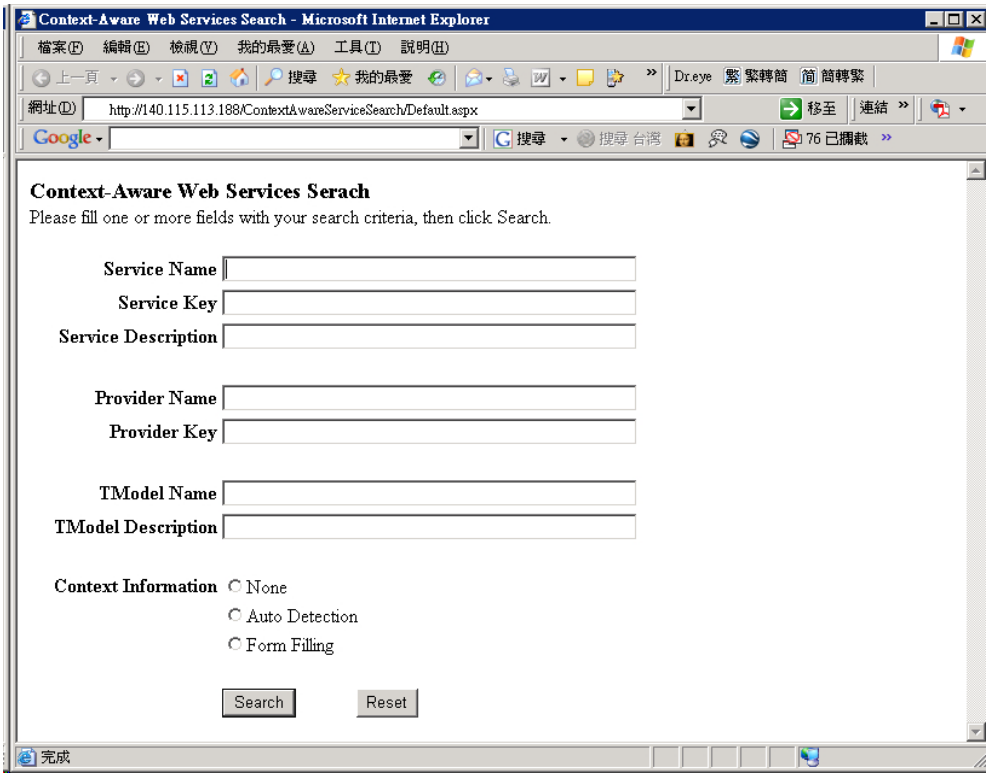

Figure 2. Interface of the request parser

A request package in OWL-S as shown below is wrapped to describe a request's contextual information. The meeting caller is *Steve* who is using either *NB* or *PDA* to access Web meeting service via *10Mbps bandwidth*. Steve's service request requires the "*availability*" of the meeting service is *99%* or above. Such a meeting requires both audio and video devices. In addition, the

service is required to support both the *meeting* mode (with *Audio Off*) and the *driving* mode (with *Video off*).

```
<?xml version="1.0"?>
<rdf:RDF>
<owl:DatatypeProperty rdf:ID="Social_Owner_Name">
   <rdfs:domain rdf:resource="#Social_Owner"/>
   Steve
   <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="Default_Value_of_Device">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
   NB, PDA
   <rdfs:domain rdf:resource="#Preferences"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="Bandwidth">
   <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
   10Mbps
   <rdfs:domain rdf:resource="#Functional_Constraints"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="Availability">
   <rdfs:domain rdf:resource="#Non-functional_Constraints"/>
   99%
   <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<Situation rdf:ID="Meeting"/> Audio Off
<Situation rdf:ID="Driving"/> Video Off
</rdf:RDF>
```

Services' Contextual Description and Wrapping

To avoid overhead caused by manual inputs of services' contextual description, we construct service agents to automatically wrap services' contextual information. Follows is a stepwise procedure to wrap services' contextual information and publish it to broker agents.

1. A service agent provides a service parser as an interface (shown in Figure 3) for guiding service providers to input services' descriptions: service name, service description, provider name, provider email, access point URL, WSDL URL, and contextual description URL.
2. The service parser transforms service description to OWL-S.
3. The service agent instantiates the service ontology to generate instances of services' static context profile.
4. The service agent wraps and packages the service along with services' contextual information for publication.
5. The service agent sends the published service to a broker agent.

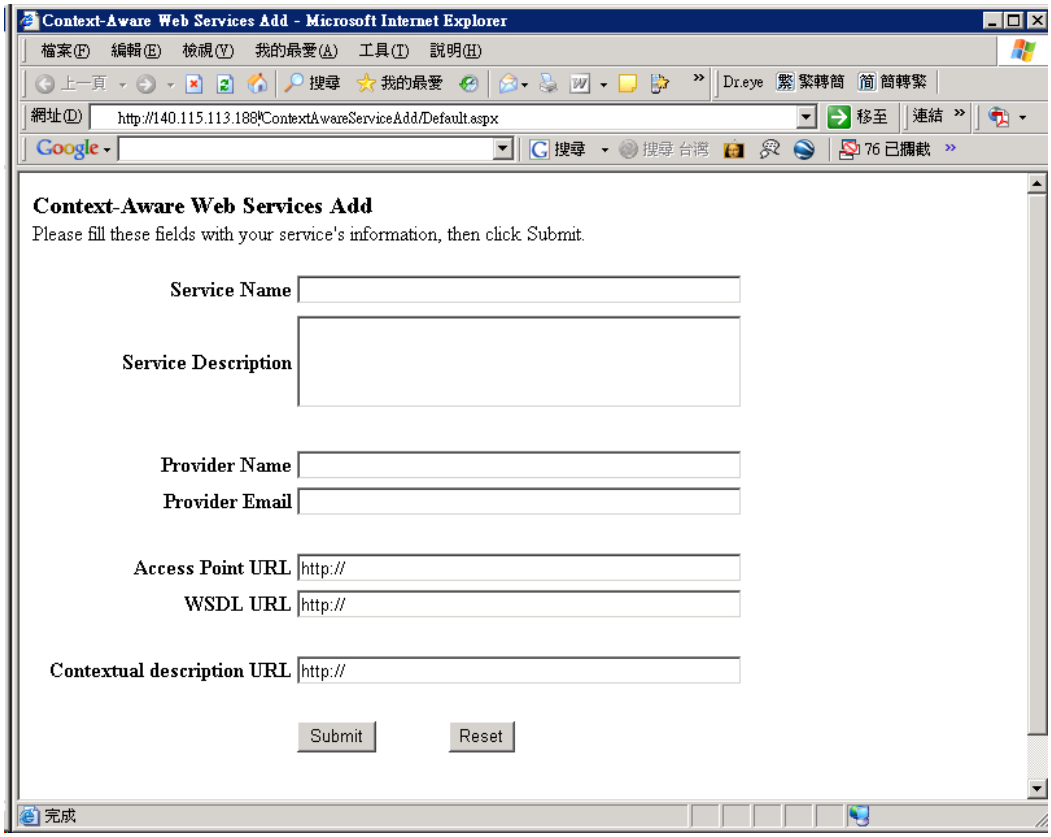

Figure 3. Interface of the service parser

The following segment of OWL-S code represents one candidate service that satisfies the above requests. It is entitled "*Web business meeting,*" which can be used by either *NB* or *PDA* devices via wireless *WLAN* or *GPRS* with *99%* availability for any requesters who are out of office.

```
<owl:DatatypeProperty rdf:ID="Service_Name">
        <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
        Web business meeting
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="Availability">
    <rdfs:domain rdf:resource="#Non-functional_Constraints"/>
    99%
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="Network">
        <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
        Wireless_LAN, GPRS
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="Platform">
    <rdfs:domain rdf:resource="#Software"/>
    NB, PDA
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="OS">
```

```
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  Windows NT, Windows XP
  <rdfs:domain rdf:resource="#Software"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="Browsers">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  Microsoft IE, Mozilla Firefox, Netscape
  <rdfs:domain rdf:resource="#Software"/>
</owl:DatatypeProperty>
```

Context-aware Services Discovery

Services discovery is a process of matchmaking between a service request and published services. Based on the aforementioned contextual description of requests and services, our context-aware services discovery utilizes capability matching and semantic matching to perform services discovery. The capability matching approach is to identify the similarities of inputs and outputs between requests and published services. Three matching degrees are defined in capability matching - exact match, plug-in match, and subsumed match. (IN_{Req} and OUT_{Req} denote the inputs and outputs of a request, while IN_{Pub} and OUT_{Pub} denote the inputs and outputs of a published service.)

1. Exact match: A published service exactly fulfills a request's requirements. Both the inputs and outputs of the published service match the request's requirements, i.e., $IN_{Pub}= IN_{Req}$ and $OUT_{Pub} = OUT_{Req}$.
2. Plug-in match: A published service sufficiently fulfills a request's requirements. The outputs of the published service provides more detailed information than the request's requirements, or the inputs of the published service require less specific information than the request's requirements, i.e., $OUT_{Req} \subset OUT_{Pub}$ or $IN_{Pub} \subset IN_{Req}$.
3. Subsumed match: A published service only partially fulfills a request's requirements. The outputs of the published service provide less information than the request's requirements, or the inputs of the published service require more detailed information than the request's requirements, i.e., $OUT_{Pub} \subset OUT_{Req}$ or $IN_{Req} \subset IN_{Pub}$.

For semantic matchmaking, we have utilized broker agents, as show in Figure 1, to handle the context-aware services discovery. Broker agents help service repositories to maintain references of service profiles published by service agents. They only keep references while the service agents keep the original service ontology. Thus, service agents need to inform broker agents whenever service profiles are updated. Broker agents also provide caching mechanisms to enhance search performance if the same service is requested by multiple requesters. Based on Figure 1, the procedure of broker agents-enabled context-aware semantic matchmaking is addressed as follows.

1. A request agent sends a request wrapped with requester's contextual information to a broker agent.
2. The broker agent forwards the request to the capability matchmaker and performs capability matching.
3. If no matched service can be found, the semantic matchmaker will decompose the request into sub-requests based on requester's contextual information, and repeat capability matchmaking in Step 2 for each sub-request.
4. The semantic matchmaker returns matched services to the broker agent.

5. The broker agent replies the matched services to the request agent.

Context-Aware Content Adaptation for Finding Right Content Presentation

In ubiquitous environments, people mostly work with portable devices along with limited computing powers, small-size screens and changing conditions. Portable devices have distinct capabilities compared with desktop computers, which divergence increases the difficulty of presenting user-friendly contents for all kinds of devices universally. Besides, the condition of people's content access is more complicated in ubiquitous environments. For example, people may need to access content while they stay outside or on a move. In addition, most current Web contents are designed for desktop computers only. The default settings and style-sheets, such as image size, font size, and layout structure, are not suitable to be presented on portable devices. As a result, a technique is needed to compose and deliver adaptive content from any platform in any format to any device through any network at anytime and at anywhere. To achieve this ultimate goal, one needs to know people's computing context (environment) and provides content adaptation based on such context.

Using the motivation example presented in Introduction section, content need to be adapted before it can be delivered and presented to the users when a Web meeting switches from PC-based to PDA-based due to the contextual changes of device and network. Content adaptation is a technique to provide the most suitable content presentation according to corresponding computing context. Our context-aware content adaptation is designed to automatically adapt Web content to various formats and to enhance Web accessibility based on users' surrounding context, especially when requesters are using portable devices in a ubiquitous environment. For example, if a requester is accessing a film while she is in a meeting, then the context-aware content adaptation service should automatically turn off the sound to avoid from making noise because it is important to remain quiet in a public situation. For another example, if the requester is accessing a film while driving, then the context-aware content adaptation service should automatically turn off the video, and leave audio only for the reason of safety. In this paper, we focus on the design of content adaptation by providing rules for transforming objects' modality and fidelity to fit users' situated environment.

A Web page comprises a set of medium objects or simply objects, which are characterized with modality indicating their types such as text, video, audio, and image. Each modality is associated with fidelity indicating objects' quality such as image resolution, color depth, and video bit-rate. In order to render the same object in various devices, content adaptation needs to perform transcoding and changing object's modality and fidelity. For example, if a mobile phone can only play image with low resolution, the fidelity of an image needs to be turned to low.

We introduce a concept of Unit of Display (UOD) to define the smallest unit that has to be displayed and adapted in a Web page in a whole. An environment specification contains the facts described by user's current contextual constraints, and the adaptation rule base contains the patterns of rules for context-oriented content adaptation. For example, if the context indicating user's situation is in a public situation, UOD with audio and video modality should be adapted by changing its fidelity to mute to remain quiet. If the context indicating user's situation is driving, then the UOD with image modality should be adapted by changing its fidelity to blank for the reason of safety.

Content adaptation rules thus can be derived from the environment specifications defined in our context model. The rules are designed as patterns for transforming objects' modality and fidelity. Table 1, 2, and 3 show three categories of adaptation rules we have derived based on users'

situated environment. For ease of representation, we present these rules in a lookup table format. Rows in each table are numbered by C, N, and D, which represent Situation, Network, and Device, respectively. As appeared on column head, the original object describes object's modality and fidelity before adaptation, while the adapted object describes object's modality and fidelity after adaptation. Object's fidelity before adaptation is defined to be original depending on how the objects are originally designed.

The results of context-aware content adaptation are shown in Figure 4. The left-hand side is Web content shown in a mobile phone without adaptation, where the requesters can only see a part of the content. In contrast, the picture shown at the right-hand side is the adapted version through context-aware content adaptation service, which provides better readability and loses no information compared with normal Web browsing.

Figure 4. A comparison of Web content access and adaptation.

Please be noted that the assumed information listed in Table 1, 2, and 3 are selected for ease of illustration. The contextual information could be much more complicated in a real world. Situation information can be further described by when, where, and what activities the person is involved, as well as location identified by GPS, sensor networks, RFID, and so on. Network information can be further described by communication protocols such as GPRS, 3G, VoIP, and WiFi. Device information can be further described with CC/PP, UAProf, and so on. We also simplified the description of modality and fidelity in this paper. In a real world, modality information can be further classified into flash and streaming media, for example; fidelity information can be further described with image types (e.g. BMP, JPEG), image resolution (e.g. 24 bit), color depth (e.g. 32bit), video bit-rate, and so on.

As shown in Table 1, a situation lookup table is based on users' access situation in terms of lighting, noise level, visibility, and access situation. The underlying concept of this lookup table is that removing unnecessary medium can save transmission time. For example, it is obvious that one cannot read video in a dark situation; nor is allowed to listen to audio in a quiet public situation. Therefore, object fidelity can be adapted to blank (by removing video and leave audio only) for darkness, or adapted to mute (by removing audio and leave video only) for quietness, as shown in S2 and S4 in Table 1, respectively. Similarly, when a user's access situation is "meeting," an object should be adapted by changing its fidelity to mute to remain quiet (as shown in S21 and S23). If a user's access situation is "driving," an object should be adapted by changing

its fidelity to blank for preventing the driver from browsing content (as shown in S22, S26, and S28).

Table 1. Situation lookup table

Rule#	original object	Situation	adapted object
S1	Object(video,original)	Situation(normal)	Object(video,original)
S2	Object(video,original)	Situation(dark)	Object(video,blank)
S3	Object(video,original)	Situation(blurred)	Object(video,bright)
S4	Object(video,original)	Situation(quiet)	Object(video,mute)
S5	Object(video,original)	Situation(noisy)	Object(video,loud)
S6	Object(audio,original)	Situation(normal)	Object(audio,original)
S7	Object(audio,original)	Situation(dark)	Object(audio,original)
S8	Object(audio,original)	Situation(blurred)	Object(audio,original)
S9	Object(audio,original)	Situation(quiet)	Object(audio,mute)
S10	Object(audio,original)	Situation(noisy)	Object(audio,loud)
S11	Object(image,original)	Situation(normal)	Object(image,original)
S12	Object(image,original)	Situation(dark)	Object(image,blank)
S13	Object(image,original)	Situation(blurred)	Object(image,bright)
S14	Object(image,original)	Situation(quiet)	Object(image,original)
S15	Object(image, original)	Situation(noisy)	Object(image,original)
S16	Object(text,original)	Situation(normal)	Object(text,original)
S17	Object(text,original)	Situation(dark)	Object(audio,original)
S18	Object(text,original)	Situation(blurred)	Object(text,bright)
S19	Object(text,original)	Situation(quiet)	Object(text,original)
S20	Object(text,original)	Situation(noisy)	Object(text,original)
S21	Object(video,original)	Situation(meeting)	Object(video,mute)
S22	Object(video,original)	Situation(driving)	Object(video,blank)
S23	Object(audio,original)	Situation(meeting)	Object(audio,mute)
S24	Object(audio,original)	Situation(driving)	Object(audio,original)
S25	Object(image,original)	Situation(meeting)	Object(image,original)
S26	Object(image,original)	Situation(driving)	Object(image,blank)
S27	Object(text,original)	Situation(meeting)	Object(text,original)
S28	Object(text,original)	Situation(driving)	Object(text,blabk)

As shown in Table 2, a network lookup table is based on network bandwidth and object size. The idea behind this lookup table is to set up a threshold of network bandwidth and object's size. The object's fidelity will be adapted to lower resolution if the transmission bandwidth is less than 2M bps and the object's size is larger than 2M bytes. For a network with downloading bandwidth less than 2M bps, content adaptation needs to be done before any object over 2M (as shown in rules N2, N4, N6, and N8 in Table 2) can be downloaded. We use Network(2M-) to denote the network bandwidth less than 2M bps, and ObjectSize(2M+) to denote object size greater than 2M bytes.

Table 2. Network lookup table

Rule#	original object	network	adapted object
N1	Object(video,original)	Network(2M+)	Object(video,original)
N2	Object(video,original)	Network(2M-) and ObjectSize(2M+)	Object(video,low_resolution)
N3	Object(audio,original)	Network(2M+)	Object(audio,original)
N4	Object(audio,original)	Network(2M-) and ObjectSize(2M+)	Object(audio,low_resolution)

N5	Object(image,original)	Network(2M+)	Object(image,original)
N6	Object(image,original)	Network(2M-) and ObjectSize(2M+)	Object(image,low_resolution)
N7	Object(text,original)	Network(2M+)	Object(text,original)
N8	Object(text,original)	Network(2M-) and ObjectSize(2M+)	Object(text,low_ resolution)

As shown in Table 3, a device lookup table is based on the types of users' devices. The idea driving this lookup table is that not every handheld device can play all types of rich media; it will save unnecessary transmission bandwidth by removing rich medium if it is un-playable on certain devices. For example, for some mobile phones with limited computing powers and rendering capability, they might not be able to play video clips. Besides, rich media take longer time to be delivered over wireless communication. As a result, if an object's modality is video, audio, or image, its fidelity will be adapted to lower resolution to be played on mobile phones as shown in rules D3, D6, and D9 in Table 3.

Table 3. Device lookup table

Rule#	original object	devices	adapted object
D1	Object(video,original)	Device(NB)	Object(video,original)
D2	Object(video,original)	Device(PDA)	Object(video,original)
D3	Object(video,original)	Device(Phone)	Object(video,low_resolution)
D4	Object(audio,original)	Device(NB)	Object(audio,original)
D5	Object(audio,original)	Device(PDA)	Object(audio,original)
D6	Object(audio,original)	Device(Phone)	Object(audio,low_resolution)
D7	Object(image,original)	Device(NB)	Object(image,original)
D8	Object(image,original)	Device(PDA)	Object(image,original)
D9	Object(image,original)	Device(Phone)	Object(image,low_resolution)
D10	Object(text,original)	Device(NB)	Object(text,original)
D11	Object(text,original)	Device(PDA)	Object(text,original)
D12	Object(text,original)	Device(Phone)	Object(text,original)

EXPERIMENTS AND DISCUSSIONS

Performance Evaluation of Context-Aware Services Discovery

We have implemented three approaches of context-aware services discovery – keyword search, capability matching (CM), and semantic matching (SM). To compare their context-aware search abilities, we constructed four similar Web-based meeting systems: Web meeting, net meting, tele-meeting, and wireless-meeting. Each application is formally specified using our context ontology. Considering the example scenario described in the Introduction section, we used the three aforementioned approaches to search for qualified Web-based meeting application for Steve. The search results are summarized in Figure 5.

We use two indexes - *Precision* and *Recall* to evaluate the performance of context-aware services provision (e.g., providing services discovery). *Precision* is the fraction of the found services that are considered as relevant; *Recall* is the fraction of the relevant services that has been found. We formally define them as follows:

$$Precision = \frac{|Ra|}{|A|}, Recall = \frac{|Ra|}{|R|}$$

where:

A contains a set of relevant services been found; |*A*| is the number of services in *A*.
R contains a set of services been found that are considered relevant; |*R*| is the number of services in *R*.
Ra contains a set of services as the intersection of the sets *R* and *A*; |*Ra*| is the number of services in *Ra*.

As shown in Figure 5, over the three services discovery approaches, the *Precision* and *Recall* of semantic matching has better performance than the others. This indicates that the services discovered by semantic matching are more relevant because these services have similar contextual descriptions, and they are more likely to be in the same context domain.

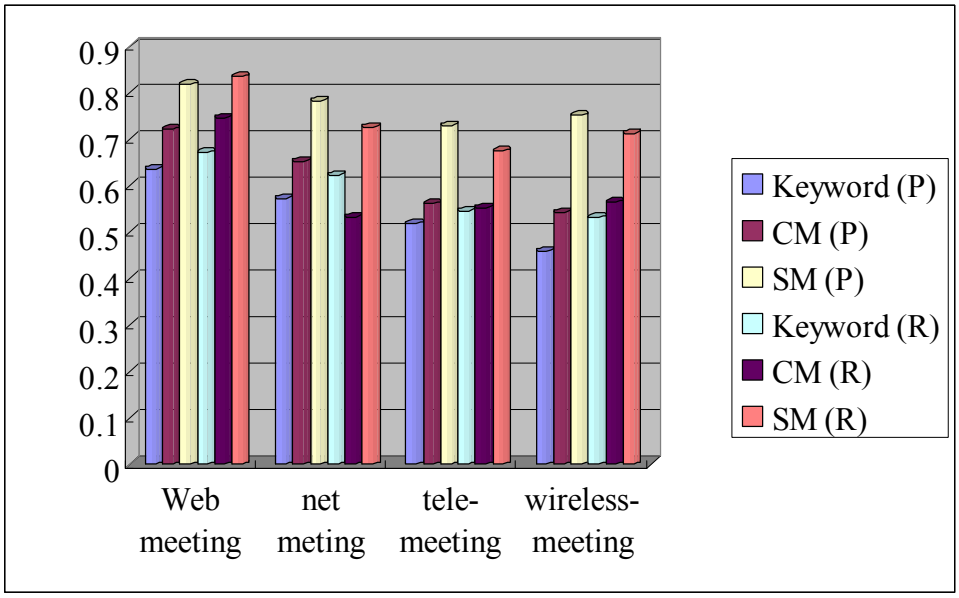

Figure 5. Precision and recalls of three different matching methods (CM: Capability matching, SM: Semantic matching)

Performance Evaluation of Context-Aware Content Adaptation

When the quality of content is degraded due to content adaptation, such degrade will affect people's comprehension of the adapted content. Thus, we have designed three questions to test requesters' comprehension.

1. Can people explain the meaning of an image appeared on an adapted page correctly? This question tests the effects of trans-coding, column-wise, and thumbnails.
2. Can people read and understand the meaning of a text appeared on an adapted page correctly? This question tests the effects of resizing, replacing, and trans-coding.
3. Can people locate a hyperlink reference appeared on an adapted page correctly? This question tests the effects of column-wise and resizing.

We designed a set of Web pages including various types of presentation information, such as text, still image, and hyperlinks. We implemented a UOD-based content adaptation engine to adapt the pages onto PDA and wireless screens. Then we hired a class of students to review the adaptation results on both PDA and wireless devices and answer the designed survey questions.

Based on students' responses, we summarize the following findings. When students use PDAs, 83% of them can correctly explain the original meaning of an adapted image. When it comes to text and hyperlink, we found that most of the students have no difficulties in understanding the meaning of adapted image. When students use mobile phones, they have difficulties in browsing content due to the smaller screen size and less computing capability, which results in lower comprehension, especially in understanding the meaning of an adapted image (67% comprehension of an adapted image).

Content comprehension	With PDA (%)	With phone (%)
image comprehension	83%	67%
text comprehension	92%	87%
hyperlink comprehension	96%	91%

CONCLUSIONS

In this paper, we have presented an integrated context model to formally define context description pertaining to service requesters and services. We designed a context acquisition mechanism with tailored environment for collecting contextual information in three phases: form-filling, context detection, and context extraction. Based on the context model, we have presented two context-aware applications for demonstrating how to find right services, right partners (collaborators), and right information (content presentation).

For finding right services, our CA-SOA differentiates from the current SOA model in several significant ways. First, ontology is enabled to bridge between service requesters, service brokers, and services. Second, ontology-based contextual repository extends the current UDDI mechanism to define services in a more comprehensive manner. Third, context-based semantic matchmaking enables more precise services discovery and access. The way how we implemented our Context Acquisition Engine using Web services leads to four major advantages: (1) the engine itself reflects the SOA concept; (2) parts or whole of the engine can be reused with flexibility and extensibility; (3) our engine has potential to be published on the Internet to facilitate context-enabled services discovery and integration; and (4) our engine can be utilized as an extension to the current UDDI engine for better service matchmaking.

For context-aware content adaptation, we use environment specifications in our context model to derive content adaptation rules into various content lookup tables, such as situation lookup table, network lookup table, and device lookup table. From the construction of the context-aware content adaptation applications, we conclude that when people use PDAs, most of them can correctly explain the original meaning of an adapted image. When it comes to text and hyperlink, we found that most of the users have no difficulties in understanding the meaning of adapted images. When people use mobile phones, they have difficulties in browsing content due to the smaller screen size and less computing capability, which result in lower comprehension especially in understanding the meaning of an adapted image.

We utilize OWL-S as a vehicle to carry contextual information. Although OWL-S is a known tool from the semantic Web society tailored for Web services descriptions, our context model and CA-SOA are not limited to OWL-S. For instance, we can use Web Services Agreement Specification (WS-Agreement) (Andrieux et al., 2006) from Global Grid Forum or Web Service Level Agreement (WSLA)(Ludwig et al., 2006) from IBM to carry the knowledge.

In our future research, we will continue to enhance our CA-SOA in the categories of rule base truth maintenance, request decomposition, services planning, and services verification. We also plan to conduct more experiments to examine performance metrics including efficiency of services composition, effectiveness of services verification, and reliability of services execution.

ACKNOWLEGMENT

This work is supported by National Science Council, Taiwan under grants NSC 94-2524-S-008-001 and NSC 95-2520-S-008-006-MY3.

REFERENCES

Abowd, G.D., Ebling, M., Hung, G., Lei, H., & Gellersen, H.W. (2002). Context-aware computing: guest editors' introduction. *IEEE Pervasive Computing*, 1(3), 22-23.

Andrieux, A., Czajkowski, K., Dan, A., Keahey, K., Ludwig, H., Nakata, T. Pruyne, J., Rofrano, J., Tuecke, S., & Xu, M. Web Services Agreement Specification (WS-Agreement). http://www.ggf.org/Public_Comment_Docs/Documents/Oct-2005/WS-AgreementSpecificationDraft050920.pdf.

Berners-Lee, T., Hendler, J., & Lassila, O., (2001). The semantic Web: A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities. *Science and Technology at Scientific American.com*. Available from http://www.scientificamerican.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21&catID=2

Broens, T., Pokraev, S., Sinderen, M., Koolwaaij, J., & Costa, P.H. (2004). Context-aware, ontology-based, service discovery. In *Proceedings of the 2nd European Symposium on Ambient Intelligence (EUSAI 2004)*, (pp. 72-83), Netherlands.

Chen G. and Kotz, D. (2000). A Survey of Context-Aware Mobile Computing Research. *Dartmouth Computer Science Technical Report TR2000-381*, (pp. 1-16).

Dey, A.K. & Abowd, G..D. (1999). Toward a better understanding of context and context-awareness. Technical Report GIT-GVU-99-22, Georgia Institute of Technology.

Khedr, M. & Karmouch, A. (2004). Negotiating context information in context-aware systems. *IEEE Intelligent Systems*, 19(6), 21-29.

Khedr, M. (2005). A semantic-based, context-aware approach for service-oriented infrastructures. In *Proceedings of 2nd IFIP International Conference on Wireless and Optical Communications Networks (WOCN 2005)*, (pp. 584-588), United Arab Emirates.

Korkea-aho, M. (2000). Context-aware applications survey. Available from *http://users.tkk.fi/~mkorkeaa/doc/context-aware.html*

Lemlouma, T., & Layaida, N. (2001). The negotiation of multimedia content services in heterogeneous environments. In *Proceedings of the 8th International Conference on Multimedia Modeling (MMM 2001)*, (pp. 187-206), Netherlands

Ludwig, H., Keller, A., Dan, A., King, R.P., and Franck, R. Web Service Level Agreement (WSLA) Language Specification. Version 1.0, http://www.research.ibm.com/wsla/WSLASpecV1-20030128.pdf

Moran, T.P. and Dourish, P. (2001). Introduction to this special issue of context-aware computing. *Human-Computer Interaction*, 16(2-4), 1-8.

Mostefaoui, S.K. & Hirsbrunner, B. (2004). Context aware service provisioning. In *Proceedings of the IEEE International Conference on Pervasive Services (ICPS)*, (pp. 71-80), Lebanon.

Mostefaoui, S.K. Tafat-Bouzid, A. & Hirsbrunner, B. (2003). Using context information for service discovery and composition. In *Proceedings of 5th International Conference on Information Integration and Web-based Applications and Services (iiWAS)*, (pp. 129-138), Indonesia.

Roman, M., Hess, C.K., Cerqueira, R., Ranganathan, A., Campbell, R.H. & Nahrstedt, K. (2002). Gaia: A middleware infrastructure to enable active spaces. *IEEE Pervasive Computing*, 1(4), 74-83.

Satyanarayanan, M. (2004). The many faces of adaptation *IEEE Pervasive Computing*, 3(3), 4-5.

Schilit, B.N., Adams, N.I. & Want, R. (1994). Context-aware computing applications. In *IEEE Proceedings of the Workshop on Mobile Computing Systems and Applications*, (pp. 85-90), USA.

Sycara, K., Klusch, M., & Lu, J., (2002), "LARKS: Dynamic matchmaking among heterogeneous software agents in cyberspace," *Kluwer Academia Publishers: Autonomous Agents and Multi-Agent Systems*, 5, 173-203.

Sycara, K., Paolucci, M., Ankolekar, A., & Srinivasan, N., (2003). Automated discovery, interaction and composition of semantic Web services. *Journal of Web Semantics: Science, Services and Agents on the WWW*, 1(1), 27-46.

Yang, S.J.H. (2006). Context Aware Ubiquitous Learning Environments for Peer-to-Peer Collaborative Learning. *Journal of Educational Technology and Society*, 9(1), Jan, 2006, 188-201.

Yang, S.J.H. Tsai, J.J.P., & Chen, C.C. (2003). Fuzzy rule base systems verification using high level Petri nets. *IEEE Transactions on Knowledge and Data Engineering*, 15(2), 457-473.

Zhang, J. Zhang, L.J., Quek, F., & Chung, J.Y. (2005). A service-oriented multimedia componentization model. *International Journal of Web Services Research*, 2(1), 54-76.

ABOUT THE AUTHOR

Stephen J.H. Yang, Ph.D., is an Associate Professor of the Department of Computer Science and Information Engineering, National Central University, Taiwan. He was the co-founder and the CEO of T5 Corp, a company providing XML-based Web services. Dr. Yang has published 2 books and over 100 technical papers in the areas of software engineering and knowledge engineering. He severed as the Program Co-Chair of IEEE MSE2003 and CAUL2006. His research interests include human computer interaction, software engineering, knowledge engineering, semantic Web, context aware ubiquitous computing, peer to peer computing, and mobile multimedia. Dr. Yang received his PhD degree in Electrical Engineering and Computer Science from the University of Illinois at Chicago in 1995. He is a member of IEEE.

Jia Zhang, Ph.D., is an Assistant Professor of Department of Computer Science at Northern Illinois University. She is also a Guest Scientist of National Institute of Standards and Technology (NIST). Her current research interests center around Services Computing. Zhang has published over 60 technical papers in journals, book chapters, and conference proceedings. She is an Associate Editor of the International Journal of Web Services Research (JWSR) and the Program Vice Chair of IEEE International Conference on Web Services (ICWS 2007 & 2006). Zhang received a Ph.D. in Computer Science from University of Illinois at Chicago in 2000. She is a member of the IEEE and ACM.

Irene Y.L. Chen, Ph.D., is an Assistant Professor of the Department of Information Management, Ching Yun University, Taiwan. Dr. Chen received her Ph.D. degree in Management of Information Systems from National Kaohsiung First University of Science and Technology, Taiwan. Her research interests include virtual community, knowledge management, and strategic management.