J Intell Inf Syst (2007) 28:105–131
DOI 10.1007/s10844-006-0005-0

Dynamic management of UDDI registries in a wireless environment of web services: Concepts, architecture, operation, and deployment

Zakaria Maamar · Hamdi Yahyaoui · Qusay H. Mahmoud

Received: 19 December 2003 / Revised: 15 May 2005 /
Accepted: 20 July 2005 / Published online: 28 December 2006

Abstract This paper presents mechanisms for the dynamic management of the content of several Universal Description, Discovery, and Integration (UDDI) registries. These mechanisms are deployed in the context of a wireless environment of Web services. By content, it is meant the announcements of Web services that providers submit to a UDDI registry. Unlike other initiatives in the Web services domain that consider a single UDDI registry and a wired communication infrastructure, this paper is concerned with the fact that: several UDDI registries are deployed, there is no wired communication infrastructure between the UDDI registries, and absence of a centralized component for coordinating the UDDI-registries. The solution presented integrates users and software agents into what we call messengers. Initially, software agents reside in users' mobile devices and cache a description of the Web services that satisfy their users' needs. Each time a user is in the vicinity of a UDDI registry, her software agent interacts with that registry, so the details stored on Web services are submitted.

Keywords Web service · UDDI · Agent · Wireless · Security

1 Introduction and motivation

Web services are emerging as a major technology for achieving automated interactions between distributed and heterogeneous applications. A Web service is an

Z. Maamar (✉)
College of Information Technology, Zayed University, Dubai, UAE
e-mail: maamarz@yahoo.com

H. Yahyaoui
Computer Sciences and Software Engineering Department, Laval University, Quebec, Canada

Q. H. Mahmoud
Department of Computing and Information Science, University of Guelph, Guelph, Canada

accessible application that other applications and humans can discover and trigger (Benatallah, Sheng, & Dumas, 2003). Various technologies back the deployment of Web services including WSDL, UDDI, and SOAP (Curbera et al., 2002). Unlike other research initiatives in the field of Web services, which consider a single UDDI registry and assume a wired and stable communication infrastructure, we are particularly concerned with the following aspects:

- Several UDDI registries are deployed across various regions. A registry is aware of the existence of other peers, but does not perform any exchange of information on its content with these peers. The exchange illustrates the mechanisms that are intended to be developed in this research. The UDDI registries may belong to different institutions, have different usage policies, and pose different requirements on acceptable announcements and retrieval requests of Web services.
- There is no pre-defined communication infrastructure between the distributed UDDI registries. An infrastructure of type wired or wireless[1] for direct interactions can be set up after assessing the importance of exchange between the UDDI registries. In addition, an UDDI registry may be withdrawn if its owner decides so.
- Absence of a central component that is responsible for managing and coordinating the UDDI registries. In a similar configuration, it is noted in Penserini, Liu, Mylopoulos, Panti, and Spalazzi (2003) that a central authority constitutes a bottleneck and may completely break down the system. On one hand, each registry is independent in defining the announcements of providers it accepts, and the retrieval requests of users it satisfies. The definition of what to accept and what to satisfy is based on a set of what we denote by *UDDI registry-defined policies*. On the other hand, each provider is independent in selecting the UDDI registries to whom it submits its announcements of Web services. The selection of where to announce is based on a set of what we denote by *provider-defined policies*.

In a Web services running-scenario, an UDDI registry takes part in two operations. The first operation is to receive the announcements of the description of Web services (also called services in the rest of this paper) from providers. The second operation is to search the registry for the services that satisfy users' needs. Examples of needs are multiple like hotel booking and car rental. The search consists of identifying the relevant services and indicating who offers them so they can be triggered after a potential composition (Casati, Shan, Dayal, & Shan, 2003). It is accepted that the advantages of Web services are due to their capacity to be composed into high-level business processes (*composite services* development). However since the announcements of services are submitted to distributed UDDI registries, this definitely leads into a different content among the registries. Thus, it is deemed appropriate to develop mechanisms for supporting the exchange of content between distinct UDDI registries.

Targeting the dynamic management of multiple UDDI registries presents some similarities with the well-known problem of information *replication* over a set of

[1] Low bandwidth, poor reliability, and absence of coverage feature this infrastructure.

distributed information sources. An immediate solution to our UDDI registry-dynamic management is to flood the communication infrastructure with the new content of any UDDI registry, which has lately been subject to changes. Changes in UDDI registries may become frequent as the number of Web services continues on growing. While the flooding fits well a wired context, the lack of reliable and permanent connections in a wireless context is a major obstacle to the flooding. Karakasidis and Pitoura (2002) have observed that the traditional database approaches of collecting, caching, and indexing data of interest in monolithic contexts becomes obsolete in global computing contexts. Hristova, O'Hare, and Lowen (2003) have made the same observation, too. Consequently, *another alternative is required to the dynamic management of UDDI registries*. In this paper, we discuss how users, while roaming, will be the vehicle of supporting the content exchange between the UDDI registries. This support is done in a transparent way because of the use of *Software Agents* (SAs). A SA is an autonomous entity that acts on the user's behalf, makes decisions, interacts with other agents, and migrates to distant hosts if needed (Jennings, Sycara, & Wooldridge , 1998). We discuss the rationale of agents in Section 2.

In this paper, each UDDI registry is associated with a structure known as *cluster* of Web services. Several clusters are made available across the wireless communication infrastructure, so providers can connect to the most appropriate one using various criteria such as cluster's proximity and workload status. The connection between providers and clusters is of type wired. However, for tracking purposes a provider cannot connect to more than one cluster, which means a provider cannot announce in the UDDI registries of multiple clusters. The cluster in which a provider posts its services for the first time is called *master*. Interesting is the situation where providers have similar services but respectively announces their services in separate UDDI registries. Benatallah et al. (2003) explain the notion of service similarity with the concept of *service communities* where alliances are formed among a potentially large number of services performing the same operation types. Unless some appropriate exchange mechanisms are set, an UDDI registry would never be aware of the existence of similar services in other registries. Besides that, for a user wishing to satisfy her needs by triggering or composing appropriate Web services, she should be given the opportunity to consider all the existing services regardless of where they are announced. These two basic cases motivate the importance of supporting a content exchange between UDDI registries.

A part of our solution to the dynamic management of UDDI registries relies on users who are mobile and have mobile devices (Abowd et al. use the term smart phone to quality the latest mobile devices and report that over a billion mobile phones exist today (Abowd, Iftode, & Mitchell, 2005)). The other part of the solution relies on software agents. It is accepted that agents are suitable candidate for performing the composition of Web services on behalf of users (Huhns, 2002; Kuno & Sahai, 2002). Thus, we integrate users and software agents into what we call *messenger*. While residing in the mobile device of a user, the software agent caches a description of the list of Web services that are involved in the satisfaction of one of the user's needs and appends this description with various details indicating for example the UDDI registry from which the agent has collected information on these Web services. On behalf of providers, users announce services in various UDDI registries to be associated with clusters denoted by *slaves*. Because users have mobile

devices, mobile support stations manage these devices in terms of identifying their physical location and handling their incoming and outgoing messages/calls (Maamar, Ben-Younes, & Al-Khatib, 2003). A mobile support station communicates with mobile users, within its radio coverage area known as *wireless cell*. For the needs of this research, each cluster of Web services is attached to a mobile support station. Therefore, When a user enters a new cell (i.e., under the coverage area of a new mobile support station) an exchange of information occurs between the agent of the user and the UDDI registry. This exchange enables to a certain extent updating this registry's content. It should be mentioned at this stage that a user does not have to visit all the clusters. Her association with a mobile support station depends on her *route* to various places like work, mall, etc.

Because a UDDI registry receives information on Web services from two independent sources namely providers of Web services, and agents of users, we decompose the services into two types: *internal* and *external*. Internal services are announced in a UDDI registry of a master cluster (providers take care of the announcements). This registry has a full control over the internal services by guaranteeing for example their QoS. And external services are always announced in an UDDI registry of a slave cluster (agents take care of the announcements). This registry cannot guarantee for example the QoS of the external services and their availability in their respective provider host for triggering purposes. External services constitute one of the challenges of managing UDDI registries.

In this research, the exchange of the UDDI registries' content does not target a total *replication*. A total replication between the UDDI registries might happen but is subject to the following two factors:

- The route of users: users are not forced to visit all the clusters so registries are fed with a new content. The participation of users in the dynamic management of the UDDI registries is not considered as a burden on them. Because of the diversity of the routes of users, the update of a UDDI registry takes place each time these users are associated with a new support mobile station of a cluster and thus, in contact with a new UDDI registry.
- The different policies that exist like UDDI registry-defined policies and provider-defined policies. For instance, each registry is independent in defining the announcements of providers it accepts and the retrieval requests of users it satisfies. Plus, each provider is independent in selecting the UDDI registries to whom it will submit its announcements of Web services.

The rest of this paper is organized as follows. Section 2 first, outlines the why of using software agents in the dynamic management of UDDI registries, and second overviews Web services and UDDI registries. Section 3 presents the agentification process of a wireless environment of Web services. Section 4 presents the security of the UDDI registries. Section 5 discusses the implementation work of this environment of Web services. Future and related work are outlined in Section 6 and Section 7, respectively. Finally, Section 8 concludes the paper. It should be noted, at this stage of the paper, that the lack of computing resources that feature some mobile devices (which is not the case with the latest developments in the field) does not prevent these devices from storing information as there will be no major computing running over them.

2 Preliminaries

Software Agents. It is a piece of software that autonomously acts to carry out tasks on the users' behalf (Jennings et al., 1998). In agent-based applications, it is accepted that users only need to specify high-level goals instead of issuing explicit instructions, leaving the decisions of how and when to their respective agent. A software agent exhibits a number of features that make it different from other traditional components including autonomy, goal-orientation, collaboration, flexibility, self-starting, temporal continuity, character, communication, adaptation, and mobility. It should be noted that not all these characteristics need to embody an agent.

Besides the availability of several approaches and technologies related to the deployment of Web services (e.g., SOAP, UDDI, Salutation), they are all tailored to a context of type wired. In a similar context, the computing resources are fixed and connected through a permanent and reliable communication infrastructure. The application of these approaches and technologies to a context of type wireless is not straightforward. Indeed, major adjustments are required because of multiple obstacles ranging from potential disconnections of mobile devices and unrestricted mobility of persons to power scarcity of mobile devices and possibility of capturing the radio signals while in the air. These obstacles highlight the suitability of software agents as potential candidates to overcome them. First, a SA is autonomous. Thus it can make decisions on the user's behalf while this one is disconnected. Second, a SA can be mobile. Thus it can move from one host to another. A continuous network connectivity is not needed (Bagci, Petzold, Trumler, & Ungerer, 2003; Bellavista, Corradi, & Stefanelli, 2002). Third, a SA is collaborative. Thus it can work with other agents that identify for example providers of Web services. Last but not least, a SA is reactive. Thus it can monitor the events that occur in the user's environment, so relevant actions can be promptly taken.

Web service. Benatallah et al. (2003) define a Web service as an accessible application that other applications and humans can discover and trigger. Plus, Benatallah et al. associate the following properties with a Web service: independent as much as possible from specific platforms and computing paradigms; developed mainly for inter-organizational rather than intra-organizational situations; and easily composable (i.e., its composition with other Web services does not require the development of complex adapters).

Universal Description, Discovery, and Integration. The UDDI specifications define a way to publish and discover information on Web services. At a conceptual level, the information provided in a UDDI business registration consists of three components (uddi.org., 2000). First, the *white pages* component includes address, contact, and known identifiers. Second, the *yellow pages* component includes industrial categorization based on standard taxonomies. Finally, the *green pages* component includes the technical information about services that a business exposes. At a business level, the UDDI business registry can be used for checking whether a given partner has particular Web service interfaces, finding companies in a given industry with a given type of service, and locating information about how a partner or intended partner has exposed a Web service. The objective is to get aware of the technical details required for interacting with that service.

3 Agentification of a wireless environment of web services

3.1 Agent-based deployment

The agentification of the dynamic management of the UDDI registries resulted in the identification of three types of agents: *provider-agent*, *UDDI-agent*, and *user-agent*. Figure 1 illustrates the different agents according to the clusters of Web services, the UDDI registries, the mobile support stations, and last but not least the wireless communication infrastructure. Usually, the coverage areas (represented by circles in Fig. 1) of the mobile support stations overlap. However to keep the figure clear the overlapping is not represented. Because the clusters of Web services are wirelessly connected (dashed lines in Fig. 1), a continuous and reliable exchange of the content of the different UDDI registries cannot take place. Figure 1 also depicts the notion of messenger, which is in fact the couple *(user,user-agent) on the move*. Users with their mobile devices are always under the management of a particular mobile support station. When a user moves to a different place, which is outside the coverage area of a mobile support station, a *handover* occurs between this station and the new mobile support station that covers this place. In the following, the rationale and role of each agent are presented.

A provider-agent identifies a provider (i.e., a business) that intends to post its Web services on the UDDI registries of the multiple clusters. However a provider is only authorized to connect to one cluster so it can use its UDDI registry. We recall that this cluster is known us master. The services announced in the UDDI registry of a master cluster are labelled as internal. This labelling has several advantages as it helps in identifying the UDDI registry where the services are announced for the first time, knowing the location of the providers to whom the services belong, denying all the operations of user-agents that aim at updating the description of the services, and naming the UDDI-agent that ensures that the parameters of the services (e.g., execution cost, execution time, QoS) are met during their execution.

Since a provider only announces its Web services in one UDDI registry, messengers take care of the UDDI registries of the remaining clusters. We recall that

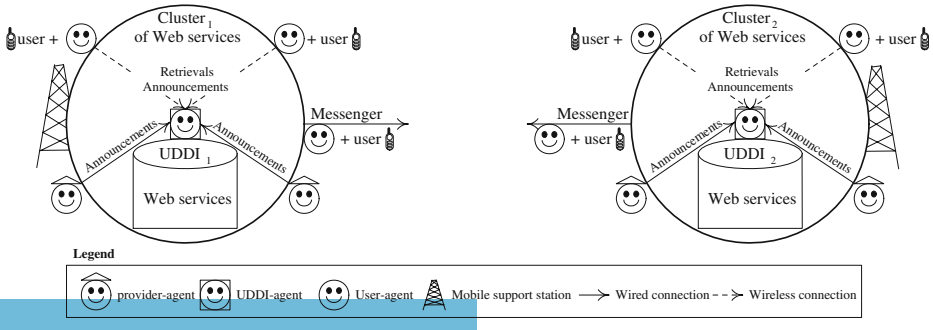

Fig. 1 Agentification of distributed UDDI registries

if	*trustworthiness*:	(provider_agent$_i$, uddi_registry$_j$) < 0.5
then	user_agent:	not(*announce*(web_service, *in*(uddi_registry$_j$)))

Fig. 2 Sample of a provider-defined policy

these clusters are known as slave.[2] For announcement purposes in slave clusters, the messengers need to comply with certain policies as it is going to be explained in the next paragraphs. The services posted on the UDDI registry of a slave cluster are labelled as external. This labelling has several advantages as it helps in (a) indicating that a third entity (i.e., a certain messenger) has announced the services, (b) informing that the services can always be subject to unpredicted changes, and (c) stating that the UDDI-agent cannot guarantee the execution parameters of the services. Wired connections support the interactions between provider-agents and UDDI-agents (Fig. 1). These interactions are for announcing new services or for updating or withdrawing the services already published. Regardless of the type of Web services whether internal or external, the labelling supports in general first, their identification and second, their tracking at the level of the UDDI registries. In addition, the labelling supports the validity of the announcements of messengers, so contradictory information are avoided (Section 3.2).

Because users are heavily engaged in announcing Web services, the agreement of the respective providers of these Web services is required. For various reasons like *privacy* (e.g., a provider does not want to announce all its services in a certain UDDI registry), *security* (e.g., a provider is afraid that its announcements of services will get altered in a certain UDDI registry), and *trustworthiness* (e.g., a provider is not confident in the security mechanisms of a certain UDDI registry), a provider has to clearly state in the description of a Web service submitted to a UDDI registry of a master cluster whether this service can be announced in the UDDI registries of slave clusters. The statements of a provider are done through a set of policies, which are appended to the announcements of services. The statements are meant for the messengers that will be roaming the different clusters. It should be noted that all the services of a provider have to be posted on at least one cluster, which is in that case the master cluster. Figure 2 contains a sample of a provider-defined trustworthiness policy: provider-agent$_i$ forbids to any user-agent the announcement of its Web services in UDDI-registry$_j$, if the trustworthiness value between this provider and this UDDI registry is less than 0.5.

Trustworthiness value calculation. The trustworthiness value of a provider of Web services towards an UDDI registry is defined by the number of times an UDDI registry suggests with success the external services of a provider (announced as internal services in other UDDI registries) to be involved in a composition process vs. the number of composition processes that are devised to satisfy users' needs (by success, it is meant services being triggered). It is noted that a UDDI-agent does not have a full control over the external services. It may happen that a service is announced in a UDDI registry of a slave cluster, but its provider has pulled out the service for maintenance.

[2] Because a cluster is seen from a provider perspective, it can be at the same time master and slave.

A user-agent resides in a mobile device (e.g., cell-phone, personal digital assistant) of a user constituting both a messenger when they move (Fig. 1). The main role of a user-agent is to satisfy its user's needs (e.g., car rental, hotel booking) after it identifies and selects the relevant Web services with the help of an UDDI registry. To this purpose, the user-agent initiates interactions with the UDDI-agent of a UDDI registry. The consideration of a specific UDDI registry depends on the current location of the user with regard to the mobile support station, which is responsible for managing her mobile device. The composition of the services may also be required to satisfy certain users' needs (e.g., travel planning requiring flight reservation, hotel booking, attraction search, user notification services) but this is beyond this paper's scope. Once the services are identified and triggered for execution, the user-agent caches in the user's mobile device various information like the identifier of the services, the UDDI registry with whom the user-agent has dealt with in order to obtain the services, and the providers of services and their location according to the master clusters. A refinement of the information that user-agents store is given in Section 3.2. This information can be potentially announced to distinct UDDI registries of slave clusters for further processing. This can be done after verifying the authorization policies of the announcements (Fig. 2). A user-agent is not authorized to update the details of any service that is announced as internal in a master cluster.

By default, users are always attached to one cluster because of the mobile support station. The type of cluster whether master or slave is not relevant. Whenever a user moves to a different place that is outside the coverage of the current support station, her mobile device becomes under the management of a new mobile support station covering this place. This means that the user-agent can now start dealing with the UDDI registry of the new cluster of Web services. The user-agent keeps track of all the clusters it has visited, its last date of visit, and the kind of information it previously submitted to their respective UDDI registry. If the user-agent notes that the information it caches is beneficial to the UDDI-agent (what it was submitted *vs.* what it can now be submitted), a wireless communication is established between both agents upon user-agent's request. The rationale of the communication is to transfer the information on the Web services to the UDDI registry so its content can get updated. The transfer has to obey to three categories of policies: provider-defined, UDDI registry-defined, and user-defined.

1. Provider-defined policies: the purpose of these policies is explained in the description of a provider-agent. A sample of a provider-defined policy is given in Fig. 2.
2. UDDI registry-defined policies: the purpose of these policies is to clarify for a UDDI-agent whether to accept announcements on external services and from which sources. A source can be either a user-agent or an UDDI registry. Figure 3 is a sample of a UDDI registry-defined policy: UDDI-agent$_i$ will not accept the announcements on Web services from any user-agent if this user-agent has collected information on these services from UDDI-registry$_j$.

```
if      source:  (user_agent, web_service) = "uddi_registry_j"
then  uddi_agent_i:  not(accept(web_service, from(user_agent)))
```

Fig. 3 Sample of a UDDI registry defined-policy

if	*authorization*:	(user_agent$_i$, uddi_registry$_j$) = "yes"
then	user_agent$_i$:	*announce*(web_service, *in*(uddi_registry$_j$))

Fig. 4 Sample of a user-defined policy

3. User-defined policies: the purpose of these policies is to define for a user-agent the UDDI registries of slave clusters where it can submit its announcements of external services. Figure 4 is a sample of a user-defined policy: user-agent$_i$ is authorized to announce its external services to UDDI-registry$_j$.

A UDDI-agent is deployed on top of an UDDI registry. The UDDI-agent interacts through wired connections with provider-agents for their announcements of services of type internal. In addition, the UDDI-agent interacts through wireless connections with user-agents for their retrieval requests of services and announcements of services of type external as well. The details on these interactions and their related policies have been explained above.

With regard to managing the UDDI registries, it may happen that the announcements of Web services are not free-of-charges mainly for external services. In that case, a provider can specify a policy to avoid the posting of services in all the UDDI registries that request for charges. In Fig. 5, the policy states the following: if provider-agent$_i$ understands that its announcements of Web services through user-agents will be charged in UDDI-registry$_j$, then this provider-agent will give up declaring in this registry. Furthermore, it may happen that for the need of refreshing the content of a UDDI registry a UDDI-agent deletes the external services from its registry without notifying the respective providers. The opposite cannot apply to the internal services as the requests of deletion only originate from their respective providers. Besides the various operations that a UDDI-agent takes care, this agent verifies the announcements so contradictory information in the content of a UDDI registry are discarded.

3.2 Messenger-based operation

We decompose the messenger operation of the wireless environment of Web services into three parts (Fig. 6). The first part consists of storing the description of the Web services that satisfy the needs of users. The second part, which is the core of this paper, consists of updating the UDDI registries according to the messenger approach. Finally, the third part consists of identifying the Web services that satisfy the needs of users considering the fact that several UDDI registries with different contents are in operation. In what follows, the second part is detailed, whereas the first and third parts are only outlined.

First of all, we assume that a description of the Web services that satisfy users' needs is already stored in their mobile devices (Part 1). This description was briefly

if	*charges*:	(provider_agent$_i$, uddi_registry$_j$) = "yes"
then	user_agent:	not(*announce*(web_service, *in*(uddi_registry$_j$)))

Fig. 5 Sample of a provider defined-policy

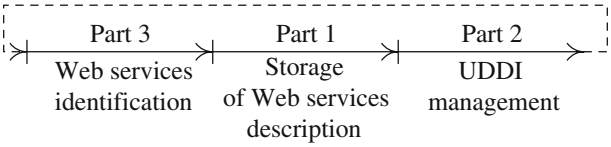

Fig. 6 Decomposition of the messenger-based operation

presented in Section 3.1. It is now refined as follows: service name (i.e., identifier), service type (internal or external), UDDI-registry identifier with regard to the master cluster where the service is announced as internal (this is extremely important for invocation purposes), UDDI-registry identifier with regard to a certain cluster from where the information on the service has been collected, outcomes of announcement policies (e.g., list of UDDI registries where a service cannot be announced), and extra details for selection purposes as providers have services in common (e.g., service execution-cost). The description of a Web service announcement that a user-agent caches is specified as a WSDL file.

When a user moves to a new coverage area, her mobile device is automatically assigned to a new mobile support station so, the device can remain reachable in terms of receiving/submitting incoming/outgoing calls/messages. It should be noted that the wireless cell of each mobile support station has an identifier, which is periodically broadcasted to the mobile devices within the cell. When a user-agent that runs on a mobile device detects a new identifier, it checks the last time it interacted with the UDDI registry that has this cell's identifier. If the user-agent has a new description of Web services compared to the previous interactions it had with the UDDI-agent of this UDDI registry, a decision to transfer this description is made. This transfer is subject to a positive verification of the different provider-, user-, and UDDI registry-defined policies. User-agents comply with the provider and user-defined policies (i.e., where to submit), whereas UDDI-agents comply with the UDDI registry-defined policies (i.e., from whom to accept).

In addition to the various policies that are integrated into the messenger-based operation, a UDDI-agent accepts announcements from messengers if-and-only-if they do not contradict the previously-made announcements. In case of contradictions, the UDDI-agent takes various measures such as discarding an announcement or holding up an announcement for further investigation. A contradiction could happen because the information that messengers transport can be subject to alteration (Section 4). In what follows, we provide a succinct list of contradictions using illustrative examples. In Fig. 7, `WS` and `P` stand for Web service and provider, respectively. In addition, contradictions are in italic.

The dots in the announcements correspond to the extra details that go along with these announcements such as the outcomes of policies, the execution cost of a service, and the reputation of a service.

1. An announcement makes a reference to a non-existing UDDI registry. We recall that the UDDI registries are aware of their mutual existence.
2. An announcement sent from a user-agent to an UDDI registry contains information on a Web service (e.g., type) that is different from the information stored in this UDDI registry. However both information are collected from the same UDDI registry. In Fig. 7a, $announce_{23}$ contradicts $announce_{22}$ since there is a difference at the level of the service type. $Announce_{23}$ and $announce_{22}$ are

A)	
UDDI$_1$:	
Master side:	Announce$_{11}$(WS$_1$, P$_2$)
Slave side:	null
UDDI$_2$:	
Master side:	Announce$_{21}$(WS$_1$, P$_1$)
Slave side:	Announce$_{22}$(WS$_1$, internal, P$_2$, UDDI$_1$, UDDI$_1$, $\cdots$)
	Announce$_{23}$(WS$_1$, *external*, P$_2$, UDDI$_1$, UDDI$_1$, $\cdots$)
B)	
UDDI$_3$:	
Master side:	null
Slave side:	Announce$_{31}$(WS$_1$, external, P$_1$, *UDDI*$_3$, UDDI$_2$, $\cdots$)

Fig. 7 Examples of contradictory announcements

respectively made by user-agent$_1$ and user-agent$_2$, whereas announce$_{11}$ is made by provider-agent$_2$ of provider$_2$.

3. An announcement on an external service refers to a provider in a UDDI registry even though this provider did not announce in this UDDI registry. Figure 7b illustrates this type of contradiction; there is no announcement from provider-agent$_1$ of provider$_1$ in the master side of UDDI registry$_3$. Announce$_{31}$ is made by user-agent$_3$.

When a UDDI-agent receives an announcement on a Web service, it checks first the type of the service. For a service of type internal, which means being posted by a provider, the UDDI-agent registers it within its UDDI registry. For a service of type external, which means being posted by a user-agent, the UDDI-agent checks if this service is already registered. If needed, new records to store announcements on Web services are created.

During the interaction sessions occurring between user-agents and UDDI-agents, several obstacles hinder the reliability and continuity of these sessions. While we provide solutions to some of the obstacles, others are challenging and require further investigation.

- Because a UDDI-agent of a UDDI registry deals with several user-agents, the UDDI-agent can easily become a single point-of-failure or a bottleneck for the registry. To address this obstacle, we suggest associating each new user-agent with a *slave-agent* that a UDDI-agent dynamically creates. A slave-agent receives announcements on services of type external, checks them according to the policies of the UDDI registry to whom these announcements are designated, checks them out to avoid contradictions, and finally updates the registry. Once the update of the UDDI registry is successfully completed, the slave-agent is automatically destroyed.
- It is known that wireless communications are less reliable than wired communications and thus, can be subject to frequent and unforeseen disconnections. A disconnection may happen while a user-agent is transferring a content to a UDDI-agent. A solution consists of repeating the transfer a certain number of

times subject to sending an acknowledgment message from the UDDI-agent to the user-agent.

– By default a user is mobile, which means she is not restricted to any specific location. While our solution to the UDDI-registry dynamic management takes advantage of the fact that users are mobile, this mobility has a side effect on the messenger-based operation. Indeed, when a transfer of information is being conducted between a user-agent and a UDDI-agent through a certain mobile support station, the user may suddenly decide to move to a different location, which is outside the coverage area of the current mobile support station. The decision of user to move has several consequences. For instance, the user-agent may not have had enough time to complete its transfer of information to the UDDI-agent. Therefore, the UDDI registry is not updated. How to handle this scenario requires further investigation.

The following running scenario is an example of the messenger-based operation. John is ready for his summer vacation and needs to do some preparation work such as booking a room in a hotel. After John submits his needs to his user-agent, the user-agent interacts with the UDDI-agent so the relevant Web services can be identified, composed according to a specific specification,[3] and finally triggered. Because most of mobile of devices are "to a certain extent" resource constrained, we propose that all the operations of satisfying needs of users take place at the level of the UDDI registry. Afterwards the results (e.g., name of hotel and flight bookings) of these operations are returned back to the user-agent prior to triggering the appropriate Web services as recommended by the UDDI-agent. Further, additional details are provided to the user-agent so it can prepare a description of the Web services that participated in the satisfaction of John's needs. Before the user-agent stores this description, it checks whether the Web services can be posted in other UDDI registries using the provider-defined policies, and interacts with John to see if he agrees on announcing these Web services too. John may not wish to reveal the Web services that were used for satisfying her summer-plan request for privacy reasons. The interaction with users aims at defining the user-defined policies for the requirement of announcing Web services in other UDDI registries. More details on a similar scenario are given in Section 5.

Several constraints hinder the process of satisfying users' needs including the *non-availability* and the *non-reliability* of the information on the Web services of type external. In the third part of the messenger-based operation, because providers will definitely have services in common, the identification of the best services whether internal or external relies on selection criteria such as execution time, execution cost, and reputation. To explain the non-availability and the non-reliability constraints, we assume that the mobile device of a user is managed by a certain mobile support station, which refers to a certain UDDI registry. When the user-agent submits its request of service retrieval to the UDDI-agent of the current UDDI registry, the following cases may happen:

[3] For the needs of our research on Web services, we use service chart diagrams for the specification of the composition process (Maamar, Benatallah, & Mansoor, 2003), but this aspect is beyond this paper's scope.

Information non-availability case: In the response that the UDDI-agent returns to the user-agent, one or several Web services required in satisfying its needs lack. In fact, the UDDI-agent may not have yet received the announcements on these Web services in its UDDI registry due to probably (a) the various announcement policies that prevent user-agents in posting Web services on this registry, or (b) the routes of users who didn't pass by the coverage area of the mobile support station of this UDDI registry. Even if the Web services that lack are announced in other UDDI registries, this does not help satisfy the needs of users.

Information non-reliability case: In the response that the UDDI-agent returns to the user-agent, all the Web services required in satisfying needs are identified. Among them, one or several services are of type external. When the time of triggering the services of type external comes, these services may not be available. Their respective providers have decided to withdraw the services for some maintenance work. This withdrawing event is only noticed in the UDDI registry of the master cluster (i.e., services announced as internal). This is not the case with the UDDI registries of the slave clusters that believe the services are still available. As a part of the solution to this problem, when services, whether internal or external, are announced, their availability periods can be indicated.

Despite that a provider submits its announcements of Web services of type internal to only one UDDI registry, this provider can track the announcements on its services now of type external at least in a part of the existing UDDI registries. Being aware of where the services are being posted may help the providers ensure a better QoS of their services in terms of availability and reliability. In fact, when a provider through its provider-agent receives a SOAP-based request to execute a service, the information on the UDDI registry, which has recommended this service can be extracted from the request. We recall that the trustworthiness value of a provider towards a UDDI registry is based on the number of requests the provider receives to perform services *vs.* the number of requests that are successfully performed (Fig. 2).

4 Security of UDDI registries

In the current operation mode of the dynamic management of the UDDI registries, connections between messengers and UDDI-agents are of type wireless (Fig. 1). Thus, the reliability of these connections has an impact on the efficiency of the information transfer to the UDDI-registries. For security reasons, the information has to be encrypted to avoid any alteration. In addition, this way of interacting with a UDDI registry assumes that the UDDI-agent can handle all the update requests that sometimes arrive at the same time from multiple messengers. A UDDI-agent can easily become a single point-of-failure or a bottleneck when a large number of users are under the management of the mobile support station that identifies the UDDI-agent.

To deal with the reliability of UDDI-agents and the efficiency of the update requests, we already overviewed in Section 3.2 a solution, which consists of deploying slave-agents. Another solution that calls for more security measures is presented as

follows and consists of two elements: enable user-agents to create *mobile delegate-agents*,[4] and deploy a *reception platform* at the level of each mobile support station. The role of the reception platform is to receive delegate-agents after their transfer from mobile devices. Before delegate-agents undertake the update of the content of an UDDI registry, they need to be controlled for various reasons as it is going to be discussed throughout this section. *Security-agents* of the reception platforms take care of the control. Once the update of an UDDI registry is completed, a delegate-agent is automatically destroyed.

As part of our general security-strategy, we discuss the mechanisms that detect malicious delegate-agents. For instance, the content of an UDDI registry can be altered by accepting announcements on non-existing services. We stated in Section 3.1 that security is one of the elements that motivate a provider to give-up from posting its services on an UDDI registry. Protecting the delegate-agents from the attacks of UDDI-agents is also important. However, this is "less" serious since delegate-agents are destroyed once they complete their operations. It is observed in Mandry, Pernul, and Rohm (2000–2001); Varadharajan and Foster (2003) that not only hosts have to be protected from malicious components, but also components from malicious hosts. Our security strategy for protecting UDDI registries includes three mechanisms: certificate verification, dynamic security-stack pool monitoring, and malicious-pattern database analysis. The first mechanism is featured by a static and dynamic analysis, whereas the remaining mechanisms are featured by a dynamic analysis.

4.1 Certificate verification mechanism

Initially, a user-agent creates a delegate-agent and associates it with a *certificate*. A certificate is generated by a certification authority and defines the actions a delegate-agent is going to perform. A certificate has a key role in our security strategy. Indeed, it is a *commitment* of what the delegate-agent is supposed to execute from the user-agent to the UDDI-agent. Any deviation from the behavior as indicated in the certificate constitutes a violation of the commitment and triggers for example an access denial to a resource. The corrective measures to take mostly depend on the severity of the violation. Before allowing a delegate-agent to perform actions, a verification of its certificate is conducted. The outcome of this verification is a classification of the security threats that originate from the delegate-agent. We classify the threats into three levels: *normal*, *suspicious*, and *dangerous*. By default, the security threat is set to normal. If the verification of a certificate confirms that the delegate-agent is malicious, then the security threat is adjusted to suspicious. A sample of the counter-measures against this threat is to block the delegate-agent by preventing the execution of its actions.

Figure 8 is a sample of a behavior specification that is included in a certificate. A certificate analysis relies on a *database of malicious patterns*. This database stores the *potential threats* to the resources of an environment of Web services. Each threat is identified with a pattern. The content of this database varies from one system to

[4] Several indicators back the impressive development happening in the field of mobile devices in terms of performance, storage capacity, and functionality (Abowd et al., 2005; Wieland, 2003; Yunos, Gao, & Shim, 2003).

```
OpenFile(x).Read(x).SendNetWork(x)
```

Fig. 8 Sample of a behavior specification stored in a certificate

another depending on the security priorities that are set for example by end-users. `OpenFile`(x)`.Read`(x)`.SendNetWork`(x) could constitute a malicious pattern if x is a secret file that is being attempted to be read and even worse to be sent through the network. Since this pattern appears in the certificate of Fig. 8, the security-agent increases the threat level from normal to suspicious. Another malicious pattern could be `While`(Exp)`.do Call` f(). Depending on the value of Exp, an infinite loop may occur. An infinite loop may block resources preventing therefore, other actions from being executed. It should be noted that we are not detecting statically infinite loops. Instead, if this pattern is identified after checking the delegate-agent's certificate, the threat level is increased from normal to suspicious. Moreover, a preventive dynamic measure could be taken. For instance, a bound on the resource use allows to deny dynamically a malicious delegate-agent from causing a denial of service. Figure 8 outlines a sample of a behavior specification stored in a certificate.

While the control of delegate-agents before their actions are executed is interesting, this control is not suitable for detecting threats that depend on dynamic information such as run-time context of a function to be called. In what follows, we discuss the second type of mechanism that our security strategy integrates.

4.2 Security stack-pool monitoring mechanism

The analysis of a behavior specification may not be enough since some called functions are only known during run-time. To deal with this limitation, we use a security stack-pool mechanism, which is a simplified version of the Java stack inspection approach (Wallach, Balfanz, Dean, & Felten, 2003). The objective is to monitor the functions by checking if a function has the right to call another one. The security-agent performs this check. At the top level of the stack, the delegate-agent's main functions are listed with their respective rights that enable accessing some specific resources. These rights are identified in a precedent authentication step that may use for example a light-authentication protocol such as KSSL (V. Gupta & S. Gupta, 2001). A function is allowed to access a resource if-and-only-if the functions it calls have the rights to access this resource. Figure 9 illustrates the security stack-pool monitoring mechanism. In this figure, F_i is a function that a delegate-agent

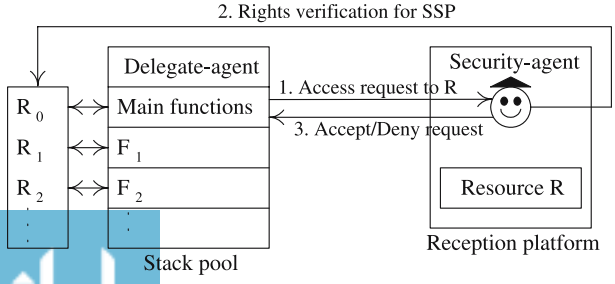

Fig. 9 Example of security stack-pool use

Input	$F_1, F_2, \cdots, F_n, R$
Get	the security context of R
If	F_i in the security context does not have the right to access R
Then	Deny access request to R
Else	Call check_rules()

Fig. 10 Validation algorithm for a resource request of access

triggers, and R_i is the right of this function F_i. F_i has the right to access a resource R if this is possible through its respective right R_i or through the rights of the functions that F_i calls.

4.3 Malicious-pattern database analysis mechanism

In Fig. 9, the security-agent of the reception platform decides whether a delegate-agent has the right to access a resource. To make this decision, two elements are required: a continuous monitoring of the functions that a delegate-agent invokes, and a verification of the security *context* of a resource R. A security context has the following format: $(F_1 \rightarrow F_2 \rightarrow \cdots \rightarrow F_n) \bullet R$ where F_i is a function called, R is a resource (e.g., data, file), $\rightarrow$ is a calling operation between functions, and $\bullet$ is an access request to the resource. The security context of a resource R is checked each time an access request to that resource is submitted. The final decision to grant or deny the access to a resource R is based on the algorithm of Fig. 10.

Check_rules() is a function that uses the database of malicious patterns. The function guarantees that the actions to be performed before granting the right to access the resource R do not constitute a malicious attack. Otherwise, the request of access is denied.

In order to check if a sequence of primitive actions (i.e., no functions called) is not malicious, a database of malicious patterns is used. The security-agent controls both the security context of the resource that is being requested for access and the primitive actions that a called function is going to perform. For illustration purposes, we consider two primitive actions a and b. When a function plans to perform a then b, the security agent checks if $a \cdot b$ does not constitute a malicious pattern ($\cdot$ represents a sequence of actions). To this end, the security-agent consults the database of malicious patterns.

Malicious patterns are detected using execution traces and data-security levels. Data are labelled with security levels such as public, shared, and secret. A malicious pattern is specified by one or several rules. Each rule states why a called function is malicious. Figure 11 is an example of a specification rule: `Read(`x`)·Send(`x`)` is a

Read·Send is a malicious pattern iff
Read: read x
Send: send x
x : secret

Fig. 11 Example of a specification rule of a malicious pattern

malicious pattern if-and-only-if Read is an action of reading a secret data x and Send is an action of sending out the same secret data x over the network.

The use of the three mechanisms namely certificate verification, security stack-pool mechanism, and database of malicious patterns depends on how much safe the delegate-agent is judged. If the delegate-agent is considered to come from a trustworthiness source,[5] then the certificate verification is sufficient. Otherwise, the other mechanisms are used. It is important to mention that these mechanisms require more computing resources but at the same time offer a better confidence in the security actions.

4.4 Learning malicious attacks

The security context of resources allows registering the malicious sequence of actions that threaten their safety. However, it would not be possible to collect in advance all the possible threats and store them in the malicious-pattern database. This database can be gradually enriched with new patterns either manually or automatically. At this time, we are adopting the manual way as the automatic one presents several challenges that we plan to address in the future. The security context of a resource contains the history of the different function calls. After a possible attack, the administrator of an UDDI registry reviews the security context. The objective is to extract the malicious sequence of actions by backtracking the security-stack pool. The administrator is assisted in her job with a dedicated tool when she stores the specification of an attack.

5 Implementation of the messenger approach

A proof-of-concept implementation of the messenger approach for the dynamic management of the UDDI registries is underway. The prototype uses Sun's Java Web Services Developer Pack 1.5 (Java WSDP 1.5), which is an integrated toolkit for building, testing, and deploying Web Services (`java.sun.com/jwsdp`). Java WSDP comes with an implementation of an UDDI registry, which we integrate into our implementation. For the client side, we use Sun's J2ME Wireless Toolkit, which provides an implementation of the Java 2 Micro Edition (J2ME) (`java.sun.com/j2me`). We have chosen to deploy our prototype on handheld wireless devices such as Palm Pilots running PalmOS. That was motivated by the fact that an access to wireless carrier equipments for testing purposes was not guaranteed. The setup of this prototype environment is as follows:

- Three 802.11 b wireless LANs are installed to cover an area of 100 m by 300 m.
- One UDDI registry is installed within the range of each wireless LAN.
- Users are equipped with PDAs, which are equipped with a wireless access card to connect to the LANs, have the MIDP4Palm implementation installed on them. MIDP4Palm is a J2ME-based Java runtime environment for PalmOS devices.

[5] Making a judgment on the trustworthiness of a source can be done off-line.

5.1 Running scenario

Initially a number of Web services are developed and registered with different UDDI registries. The UDDI-agent is implemented as a Java-based server that interacts with the UDDI registry and with user-agents. A user-agent is implemented as a networked MIDlet that can send messages across the network. Figure 12 presents multiple screen shots of the proof-of-concept. Our running scenario is a user who needs a print service that would allow her to find the closest printer. The steps that outline the messenger approach are as follows:

1. The user downloads and installs a MIDlet interface that allows her to interact with the UDDI registries and providers as well. Both are spread across different mobile support stations.
2. Once the MIDlet interface is installed (Fig. 12a), this allows the user to search for a Web service within a specific category, e.g., Mobile Office.
3. Afterwards, the user-agent communicates with the UDDI-agent within its wireless range in order to search for the Web services in the Mobile Office category (Fig. 12b).
4. If the search is successful, the UDDI-agent transfers a list of Web services to the user-agent so the user can select the appropriate service for example `PrintService` (Fig. 12c).
5. After the user-agent stores a description on `PrintService`, a request triggering that service is submitted from the user-agent to the provider of this service so a search for printers within close proximity to the user is carried out. The outcome

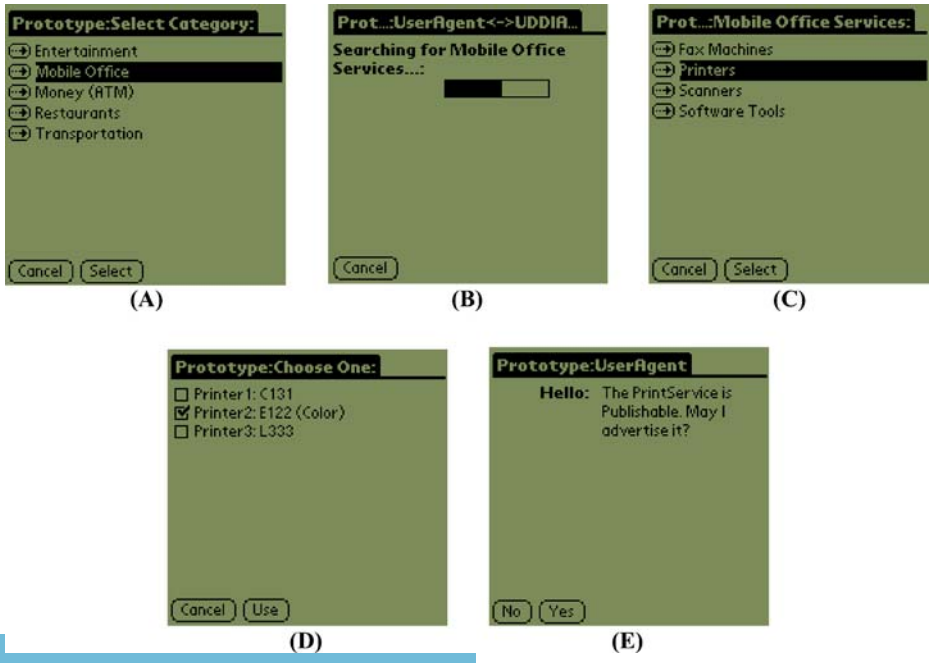

Fig. 12 Screen shots of the proof-of-concept

of this search is a list of printers from which the user selects one (Fig. 12d). The user may now remotely access the printer by clicking on the file to be sent out for printing.

6. Once the user has finished using `PrintService`, the user-agent checks if the UDDI registry and the respective provider of `PrintService` allow the posting of this service in other UDDI registries. If yes, the user-agent checks with the user whether she too agrees on publishing `PrintService` in other UDDI registries (Fig. 12e).

5.2 Performance evaluation

As mentioned earlier, the environment we used for experimentation consisted of three 802.11 b wireless LANs covering an area of 100 m by 300 m. We used three UDDI registries and eight users. Users were moving at various speeds; the speed of the users varied between 1 m/s (the user is walking slowly) and 5 m/s (the user is running). We have evaluated the performance of our solution using two types of evaluations:

- The average number of messages for registration, discovery, and copying of service descriptions. This criterion helps examine the communication overhead of the proposed solution in terms of the number of messages used for registering advertisements into UDDI registries by service providers, discovering services, and messages between user-agents and UDDI-agents to synchronize UDDI registries.
- Ratio of successful service discovery. This shows the probability for a user to discover a desired service and the probability that user-agents successfully copy services from one UDDI registry into another. This measure will demonstrate the efficiency of the proposed solution. We calculate the ratio by taking the number of successful service discoveries for a device and dividing it by the total number of discovery attempts.

To capture the effect that multiple users and UDDI registries have on the system, UDDI registries periodically receive advertisements and respond to service discovery requests. And, users periodically make service discovery requests, and their agents communicate with UDDI agents to register services. Based on the experiments, we found that: (a) the communication overhead increases as the

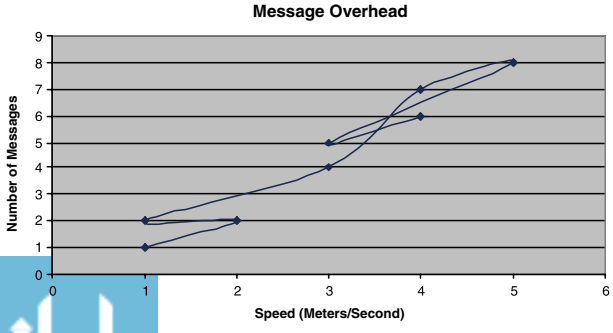

Fig. 13 Message communication overhead

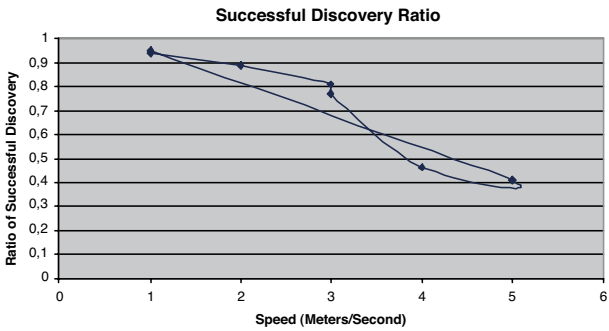

Fig. 14 Successful service discovery rate

number of users and their mobility increase. In addition, the number of users is more dominant factor in increasing the overhead especially when the users are highly mobile since their user-agents will attempt to discover UDDI registries (Fig. 13); and (b) the success ratio is higher in the case of a user moving at a slow speed, and it deteriorates faster when the user is running at the maximum speed (5 m/s) as shown in Fig. 14. This is so when users are leaving one ad hoc network and joining another.

6 Future work

The integration of messengers into the dynamic management of UDDI registries opens up the opportunity of conducting further research in the future. We identified two initiatives. The first one uses the features of ad-hoc *networking* to support the dynamic management of UDDI registries. And the second one embeds *contextual* information into UDDI registries and messengers. We detail the first initiative and briefly present the second initiative.

6.1 Support of ad-hoc networking to messengers

An ad-hoc (or "spontaneous") network is a local area network or other small network, especially one with wireless or temporary plug-in connections, in which some of the network devices (sometimes mobile) participate in the network only for the duration of a communication session, or because some devices are in close proximity so a communication session can take place.

According to Ishibashi and Bouataba (2005), mobile devices in a mobile ad-hoc network have a different role than in a conventional local-area network. In this latter type of network, communications are centered around base stations. The infrastructure up to a base station is mostly fixed, so the topology is stable. Some elements of Fig. 1 are a good illustration of this type of network and have to a certain extent framed the operation of messengers. In an ad-hoc network, mobile devices act not only as end-systems, but also as routing devices. In the following, we aim at looking into the value-added of motivating messengers to engage in ad-hoc collaboration. We first, discus the rationale of this collaboration and the challenges it faces and second, present the appropriate mechanisms we envision for carrying out this collaboration.

The mutual awareness of messengers enables emerging an ad-hoc network upon which mechanisms for conducting collaboration operate. As a prerequisite to this collaboration, it is assumed first, that messengers have obtained the necessary authorizations to collaborate from their respective users and second, that messengers are in the vicinity of each other. Currently routes of users and policies limit the messengers in their exchange of service descriptions with UDDI registries. This exchange can be boosted by allowing messengers to collect further descriptions from their peers. It is expected that this collection does not violate the multiple policies, which frame the operation of the messengers.

In Fig. 1, the current scenario for data management among distributed UDDI registries highlights a messenger that conveys information that are related to a single user. The scenario that is targeted through the ad-hoc collaboration is to enable messengers to convey information that are related to multiple users. In Fig. 15, $\text{messengers}_{1,2,3}$ are in the vicinity of each other and constitute an ad-hoc network. After exchanging information, each messenger possesses now information on the other two messengers in terms of service descriptions. For example messenger_3 can now post additional service descriptions on an UDDI registry and these descriptions are of two types, those by default associated with user_3 and those associated with $\text{users}_{1,2}$. Links between users' mobile devices are formed within direct communication range, and devices and links combine to create the network topology. During the lifetime of an ad-hoc network, messengers may move around within the network, altering the topology by creating or breaking links between devices. In addition messengers may also enter or leave the network.

Multiple aspects need to be investigated when messengers are engaged in ad-hoc collaboration including:

1. What are the mechanisms that need to embed users' mobile devices, so that messengers can detect the opportunity of collaborating independently of the opportunity of detecting their vicinity?
2. What are the steps that messengers perform once they agree on collaborating, what are the steps to conduct in order to join/leave an existing ad-hoc collaboration of messengers without disrupting the information exchange, and what are the incentives for example for joining in such a network and sharing descriptions of Web services?

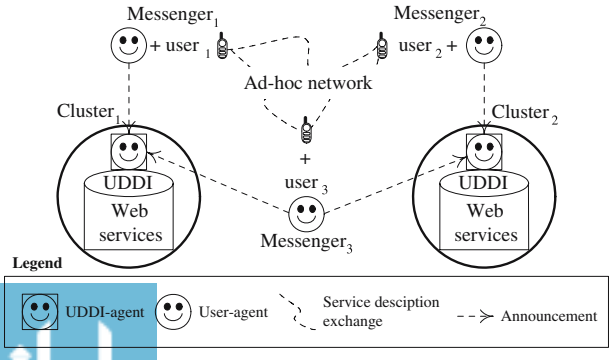

Fig. 15 Support of ad-hoc networking to messengers initiative

3. Is there a need to distinguish the description of a Web service, which is directly submitted to UDDI registries (Messenger$_i$: UDDI-Registry) *vs.* the description of a Web service, which is indirectly submitted to UDDI registries (Messenger$_i$, $\cdots$, $\rightarrow$ Messenger$_j$: UDDI-Registry)?
4. How to control/limit the number of participation of messengers in an ad-hoc collaboration, what are the reasons of this control, and how to avoid the overloading of messengers with regard to their computing and storage capacities?
5. What are the benefits in terms of just to cite a few performance and efficiency that ad-hoc collaboration caters to the flow of service description between messengers and UDDI registries?

6.2 Context into UDDI registries and messengers

Brézillon defines in context as the information that characterizes the interaction between humans, applications, and the surrounding environment. Many researchers have attempted defining context using examples. For instance, Schilit, Adams, and Want (1994) decompose in context into three categories: computing context (e.g., network connectivity, communication cost), user context (e.g., user's profile, location), and physical context (e.g., lighting, noise levels).

Our initiative on embedding contextual information into messengers and UDDI registries is motivated by the fact that different factors can contextualize the information exchange, which is expected to occur between UDDI registries and messengers. Among these factors, we cite location of exchange, period of exchange, and violation of policies. Because of the contextual information that will embed UDDI registries and messengers, several types of questions can be handled such as which messengers are attached to which UDDI registry? What services is a messenger submitting to a UDDI registry? And, what routes has a messenger passed by? We plan defining two types of context (Fig. 16): $\mathcal{U}/\mathcal{M}$-context standing for context of UDDI registry/context of messenger.

Managing contextual information becomes crucial in mobile computing scenarios. In such scenarios, it was suggested to split contextual information into three levels (Bellur & Narendra, 2005): environmental level: enables defining the overall

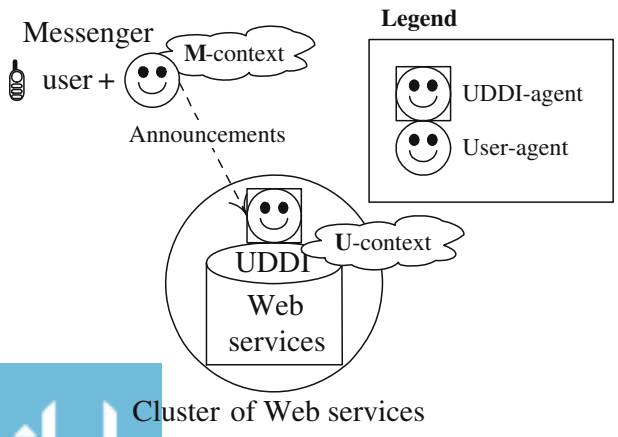

Fig. 16 Context into UDDI registries and messengers initiative

environmental context like types of mobile devices, locations, and weather conditions; service level: models and manages the context surrounding individual services offered over mobile devices; and resource level: presents the context of the resources on which the services are to operate.

7 Related work

Scenarios where people while on the move electronically interact with their surrounding environment have been reported in José, Moreira, and Rodrigues (2003); Ratsimor, O., Joshi, A., Finin, and Yesha (2003). This backs our solution of getting users transparently involved in various operations among them the update of UDDI registries. An interesting use of mobile devices is reported in Parikh (2005). Mobile devices permitted supporting the training of people who live in rural areas in India through low-cost information services. These people are not knowledge workers, do not use modern information technology resources, and would like maximal utility from computing devices with minimal engagement (using for example software agents to act on their behalf). By understanding the needs of users and their mobility patterns, it would be possible to first, offer customized services and second, take advantage of these patterns to engage users in various operations. Kim and Kotz (2005) have aimed in at classifying the mobility of users using the argument that providing QoS services requires predicting user mobility. Our work is at the crossing point of several research initiatives on Web services, UDDI registry, and wireless. While these concepts are being independently studied (except for Web services and UDDI), we aim at their combination in the same framework. In what follows, we discuss the most related initiatives to our.

In Valavanis, Ververidis, Vazirgiannis, Polyzos, and Norvag (2003), MobiShare project provides a middleware system for offering ubiquitous connectivity to mobile devices. A mobile device is seen as a source of services, a requestor of services, or both. By service, it is meant in MobiShare the data that devices decide to publicly make available. Data availability depends on the status (on/off) and location (dependent on the coverage area) of a device. While in MobiShare the devices can act as providers of services, our devices have a different role. Indeed they help announce the services of providers in different locations (i.e., UDDI registries). Users make announcements on behalf of providers is seen as a "favor" and not as a "commitment."

Service announcement and availability are another difference between our work and MobiShare. In MobiShare, each time a device moves from cell A (similar to a cluster) to cell B, the whole description of the services in the device moves also from cell A to cell B. A copy of this description remains in cell A, with a mention that the device is off-line (since it is outside the coverage area of cell A). Therefore, the description of the services is the same in cell A and cell B. In our work, the content of the UDDI registries after announcing services may be different for various reasons: each cluster has its own policy for accepting announcements of Web services from devices, each provider of services has the opportunity to decide where to announce its services, each user decides if he would like to volunteer as a messenger, and the number of users that transit by the coverage area of a cluster so they can make announcements. To conclude our comparison, a service in MobiShare can become

unavailable because a device is no longer accessible from a certain cell. In our work the availability of the services does not depend on devices. A decision on the availability of a service strictly goes back to providers.

In DBGlobe project, the aim is the development of data and metadata management techniques to deal with the challenge of global computing using a data-centric approach (Karakasidis & Pitoura, 2002). DBGlobe considers mobile entities as primary data stores and broadens the data management focus to address various issues such as mobility, autonomy, and scalability. To make data widely available, DBGlobe relies on chained hierarchies of directories. In case of an unsatisfied request at the level of a directory, the request is forwarded to a higher geographical authority. In our work, there are neither authorities nor hierarchies. All Web services clusters are put at the same level. Furthermore, clusters are totally independent in managing their respective UDDI registry.

In the eNcentive project (Ratsimor et al., 2003), peer-to-peer electronic marketing in mobile ad-hoc environments is studied. eNcentive employs an intelligent marketing scheme, by providing users the capacity to collect information like sales promotions and discounts. The marketing scheme relies on users who propagate promotions and discounts to other users with same interests and preferences in the network. Users participating in eNcentive take advantage of circulating broadly announcements since businesses that originally issued the announcements reward the active distributors with additional promotions and other compensations. Several commonalities exist between eNcentive and messenger projects. First, we both promote the idea of getting users actively engaged in information disseminating operations, independently of the information type. While users in eNcentive interact with each other, users in the messenger project primarily interact with UDDI registries, and if permitted with other users as discussed in Section 6.1. We recall that different types of policies regulate the interactions in the messenger project. Second, we both consider agents as a major concept in the design and development of our respective systems. Last but not least, we both are concerned with the issues of privacy and trustworthiness. In spite of these similarities, we strengthen the complexity of the operations that take place in the messenger project, as the aim of these operations is to support the content management of distributed UDDI-registries, whereas in eNcentive the aim of the operations is to do some information broadcasting.

The Wireless environment of Web services that is analyzed in this paper has several common features with those of peer-to-peer environments. Indeed, there is no-centralized component that is in charge of managing and coordinating the UDDI registries. In addition, all the UDDI registries have equivalent capabilities and responsibilities in terms of accepting the announcements of Web services and satisfying the retrieval requests of Web services. Because the communication infrastructure that is deployed in this paper is of type wireless, the flooding solution was abandoned and the use of messengers was promoted. This contradicts what is happening in peer-to-peer environments as observe in Castano, Ferrara, Montanelli, Pagani, Rossi (2003). The solutions proposed for content retrieval in peer-to-peer environments often exploit either flooding or broadcasting to disseminate the queries when the precise located of a searched content is unknown.

Finally, Verma et al. (2003) observed that the new version of the UDDI specification recognizes the need for the existence of multiple registries and the need for interactions among them. They noted that the challenge of dealing with hundreds

of registries (if not thousands) during service publication and discovery becomes critical. Inline with the possibility of running several UDDI registries, Budak Arpinar et al. noted that using emergent peer-to-peer computing techniques a single UDDI registry can be moved from its centralized nature to a distributed one (Budak Arpinar, Aleman-Meza, Zhang, & Maduko, 2004). This can allow service providers to select any particular registry in which their Web services would be listed. It happens for instance that competitors do not to be listed in the same registry.

8 Conclusion

In this paper, we presented our research on the dynamic management of several UDDI registries deployed on top of a wireless environment of Web services. Our solution has relied on the fact that users are mobile as well as on the latest developments happening in the field of mobile devices (more storage capacity, more computing resources, and more advanced features). To manage the content of the UDDI registries, different types of policies have been put forward stating for example where to announce, what to announce, and what to accept. These policies have allowed considering several aspects in the announcement of Web services such as security, privacy, and trustworthiness. Because of the wireless communication infrastructure connecting the UDDI registries, a flooding-based solution has been discarded. Acting as messengers, users and their agents support the exchange of content between the UDDI registries.

References

Abowd, G., Iftode, L., & Mitchell, H. (2005). The smart phone: A first platform for pervasive computing (Guest editors' introduction). *IEEE Pervasive Computing, 4*(2), 18–19. (April–June)

Bagci, F., Petzold, J., Trumler, W., & Ungerer, T. (2003). Ubiquitous mobile agent system in a P2P-network. In *Proceedings of the System Support for Ubiquitous Computing Workshop at the Fifth Annual Conference on Ubiquitous Computing (UbiComp'2003), Seattle, Washington*. Berlin Heidelberg New York: Springer.

Bellavista, P., Corradi, A., & Stefanelli, C. (2002). The ubiquitous provisioning of internet services to portable devices. *IEEE Pervasive Computing, 1*(3), 81–87. (July/September)

Bellur, U., & Narendra, N. C. (2005). Towards service orientation in pervasive computing systems. In *Proceedings of the International Conference of Information Technology Code and Computing(ITCC'2005), Las Vegas, USA*, (pp. 289–295). Washington, DC: IEEE Computer Society.

Benatallah, B., Sheng, Q. Z., & Dumas, M. (2003). The Self-Serv environment for web services composition. *IEEE Internet Computing, 7*(1), 40–48. (January/February)

Brézillon, P. (2003). Focusing on context in human-centered computing. *IEEE Intelligent Systems, 18*(3), 62–66. (May/June)

Budak Arpinar, I., Aleman-Meza, B., Zhang, R., & Maduko, A. (2004). Ontology-driven web services composition platform. In *Proceedings of the IEEE International Conference on E-commerce Technology (CEC'2004) San Diego, USA*(pp. 146–152). Washington, DC: IEEE Computer Society.

Casati, F., Shan, E., Dayal, U., & Shan, M.-C. (2003). Business-oriented management of web services. *Communications of the ACM, 46*(10), 55–60. (October)

Castano, A., Ferrara, S., Montanelli, S., Pagani, E., & Rossi, G. P. (2003). Ontology-addressable contents in P2P networks. In *Proceedings of the First Workshop on Semantics in Peer-to-peer and Grid Computing held in Conjunction with the Twelfth International World Wide Web Conference (WWW'2003) Budapest, Hungary* (pp. 55–68). New York, NY: ACM.

Curbera, F., Duftler, M., Khalaf, R., Nagy, W., Mukhi, N., & Weerawarana, S. (2002). Unraveling the web services web: An introduction to SOAP, WSDL, and UDDI. *IEEE Internet Computing, 6*(2), 86–93. (March/April)

Gupta, V., & Gupta, S. (2001). KSSL: Experiments in wireless internet security. Technical report, Sun Microsystems, SMLI TR-2001-103.

Hristova, N., O'Hare, G. M. P., & Lowen, T. (2003). Agent-based ubiquitous systems: 9 lessons learnt. In *Proceedings of the System Support for Ubiquitous Computing Workshop at the Fifth Annual Conference on Ubiquitous Computing (UbiComp'2003) Seattle, Washington*. Berlin Heidelberg New York: Springer.

Huhns, M. (2002). Agents as web services. *IEEE Internet Computing, 6*(4), 93–95.(July/August)

Ishibashi, B., & Bouataba, R. (2005).Topology and mobility considerations in mobile ad hoc networks. In *Ad Hoc Networks Journal*. (Elsevier)(in press)

Jennings, N., Sycara, K., & Wooldridge, M. (1998). A roadmap of agent research and development. *Autonomous Agents and Multi-agent Systems, 1*(1), 7–38. (Kluwer Academic Publishers)

José, R., Moreira, A., & Rodrigues, H. (2003). The AROUND architecture for dynamic location-based services. *Mobile Networks and Applications, 8*(4), 377–387. (Kluwer Academic Publishers)

Karakasidis, A., & Pitoura, E. (2002). DBGlobe: A data-centric approach to global computing. In *Proceedings of the 22nd International Conference on Distributed Computing Systems Workshops (ICDCSW'2002) Vienna, Austria*(pp. 735–740). Washington, DC: IEEE Computer Society.

Kim, M., & Kotz, D. (2005). Classifying the mobility of users and the popularity of access points. In *Proceedings of the International Workshop on Location-and Context-awareness (LoCA'2005) held in conjunction with the Third International Conference on Pervasive Computing(PERVASIVE'2005)X Munich, Germany*(pp. 19–24). Berkeley, CA: USENIX.

Kuno, H., & Sahai, A. (2002). My agent wants to talk to your service: Personalizing web services through agents. Technical Report HPL-2002-114, HP Laboratories, Palo Alto, CA USA.

Maamar, Z., Ben-Younes, K., & Al-Khatib, G. (2003). Scenarios of supporting mobile users in wireless networks. In *Proceedings of the Second International Workshop on Wireless Information Systems (WIS'2003) held in conjunction with the 5th International Conference on Enterprise Information Systems (ICEIS'2003) Angers, France* (pp. 12–20). Angers, France: ICEIS.

Maamar, Z., Benatallah, B., & Mansoor, W. (2003). Service chart diagrams—Description & application. In *Proceedings of the Alternate Tracks of the Twelfth International World Wide Web Conference (WWW'2003) Budapest, Hungary*. New York, NY: ACM.

Mandry, T., Pernul, G., & Rohm, W. R. (2000–2001). Mobile agents on electronic markets—Opportunities, risks, agent protection. *International Journal of Electronic Commerce, 5*(2), 47. (Winter).

Parikh, T. S. (2005). Using mobile phones for secure, distributed document processing in the development world. *IEEE Pervasive Computing, 4*(2), 74–81. (April–June)

Penserini, L., Liu, L., Mylopoulos, J., Panti, M., & Spalazzi, L. (2003). Cooperation strategies for agent-based P2P systems. *Web Intelligence and Agent Systems: An International Journal, 1*(1), 3–21. (IOS Press)

Ratsimor, O., Joshi, A., Finin, T., & Yesha, Y. (2003). eNcentive: A framework for intelligent marketing in mobile peer-to-peer environments. In *Proceedings of the 5th International Conference on Electronic Commerce (ICEC'2003) Pittsburgh, PA, USA*(pp. 87–94). New York, NY: ACM.

Schilit, B., Adams, N., & Want, R. (1994). Context-aware computing applications. In *Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications, Santa Cruz, California, USA*(pp. 85–90). Washington, DC: IEEE Computer Society.

uddi.org. UDDI Technical White Paper. http://www.uddi.org/,(2000). (Visited August 2003).

Valavanis, E., Ververidis, C., Vazirgiannis, M., Polyzos, G. C., & Norvag, K. (2003). MobiShare: Sharing context-dependent data & services from mobile sources. In *Proceedings of the 2003 IEEE/WIC International Conference on Web Intelligence (WI'2003) Halifax, Canada* (p. 263). Washington, DC: IEEE Computer Society.

Varadharajan, V., & Foster D. (2003). A security architecture for mobile agent-based applications. *World Wide Web: Internet and Web Information Systems, 6*(1). (Kluwer)

Verma, K., Sivashanmugam, K., Sheth, A., Patil, A., Oundhakar, S., & Miller, J. (2003). METEOR-S WSDI: A scalable infrastructure of registries for semantic publication and discovery of web services. *Journal of Information Technology and Management, 6*(1), 17–39. (also appeared as Technical Report 03-006, LSDIS Lab, Computer Science Department, University of Georgia, February 2003).

Wallach, D., Balfanz, D., Dean, D., & Felten, E. (2003). Understanding java stack inspection. In *IEEE Symposium on Security and Privacy, Oakland, California, USA*(pp. 52–63). Washington, DC: IEEE Computer Society.
Wieland, K. (2003). The long road to 3G. *International Telecommunications Magazine, 37*(2). (February)
Yunos, H. M., Gao, J. Z., & Shim, S. (2003). Wireless advertising's challenges and opportunities. *IEEE Computer, 26*(5), 30–37. (May)