Presence-Aware Infrastructure using Web services and RFID technologies

Clemens Kerer, Schahram Dustdar, Mehdi Jazayeri,
Danilo Gomes, Akos Szego, Jose A. Burgos Caja
Distributed Systems Group, Information Systems Institute
Vienna University of Technology
{kerer | dustdar | jazayeri | danilo | akos | raul @infosys.tuwien.ac.at }

Abstract. Web services have the potential to serve as a key enabling technology for environments facilitating collaborative scenarios. In this paper we analyze a same room/same time collaborative scenario, a program committee (PC) meeting. We focus on design and implementation of a service-oriented approach for PC meetings. Our proposed solution includes various Web services for the actual support of the work, Web applications combining those services, and presence-aware technology, in our case RFID (Radio Frequency Identification) tags. The contribution of this paper is to show how a presence-aware infrastructure comprising Web services and RFID technology can be built and how they provide support for the envisaged scenario.

Keywords: Web services, RFID, Presence-awareness, Service-oriented Architecture, UDDI

1 Introduction

As teams of knowledge workers have become more dispersed and mobile, the processes they are engaged in have increased in complexity and they have been inundated with various types and quantities of information. A typical knowledge worker has to carry several gadgets for communication and data management. The data and communication are used for organizing personal tasks or coordinating tasks with colleagues. *Ad hoc* team formations are an increasingly common form of cooperation for knowledge workers. Ad hoc teams are formed to solve a specific problem and have a limited lifetime. Many processes such as software inspections and reviews or task forces rely on such teams. Members of ad hoc teams often come from different organizations and different locations. The members share information and interact for the limited lifetime of the project. Members are often mobile and meetings may be held physically or virtually. Such teams traditionally use separate applications for different parts and phases of the project.

An example of such an ad hoc team is a conference program committee. We present a scenario of a conference program committee process as a driving example. We use the example to show the many different tasks and processes involved in order to be able to identify where environment services based on Web services and RFID (Radio Frequency

Identification) technology may be helpful. A conference program committee (PC) is formed to solicit and select papers to be presented at a conference. A conference's steering committee appoints a PC Chair to run the process of paper selection. The chair then selects members of the program committee. Typically, the committee is selected from experts in the field using various criteria to ensure that the committee includes a broad mixture of subspecialties, a balance of academia and industry, and geographical diversity. The major milestone for the committee is the program committee meeting in which the submitted papers are discussed and the best ones selected for conference presentation. The meeting takes place over one to two days and involves considerable cost and effort due to travel and local arrangements. There are, however, many pre- and post-meeting activities as well, just as there are for any review meeting processes. For the program committee process, we identify pre-meeting, meeting, and post-meeting phases [12]. In this paper we focus on the meeting phase and the role of Web services and RFID.

Web services have the potential to serve as an environment facilitating collaborative scenarios such as PC meetings. In this paper we focus on design and implementation of a service-oriented approach for PC meetings. Our proposed solution includes various Web services for the actual support of the work and also presence-aware technology, in our case RFID tags.

The contribution of this paper it to show how a presence-aware infrastructure comprising Web services and RFID technology provides support for a same time/same room scenario such as the program committee meeting.

The reminder of the paper is organized as follows: Section 2 provides a brief overview on the architectural considerations for the presence-aware infrastructure for program committee meetings. Section 3 discusses design and implementation of our DMC Lab (Distributed and Mobile Collaboration) core services. Section 4 analyzes the current prototype and discusses our future work. Finally, section 5 concludes the paper.

2 Architectural considerations

The architecture used for our services follow the Service-oriented Architecture (SOA) style [2, 3] depicted in Figure 1. We distinguish between (a) system level services and (b) application level services. *Core services* comprise services which are required for almost all other services and include directory services (in our case LDAP), a UDDI service registry [1, 4, 5], and an RFID service. *Application services* consist of a combination of other services. The reason for this is that the service requestors may include a multitude of mobile devices (e.g. PDAs, mobile phones, laptops) with no other software installed except a browser.

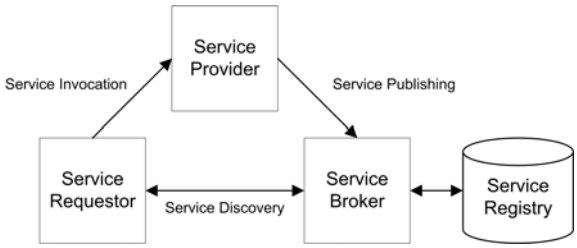

Fig. 1. Service-oriented Architecture

The Web applications themselves represent the combination of existing services. We distinguish between static and dynamic service composition. For static composition a Web application controls the service invocations and flows. It may be either "hard wired" inside the Web application code or using a BPEL (Business Process Execution Languages for Web services) [10] process description. But in both cases the modeling of the service flows has to be done in advance, i.e. before invocation. Dynamic composition in our context has the notion of associating and aggregating many services dynamically. Figure 2 shows one case when a Web application directly or indirectly uses services {AS1, CS1, CS2}. The control and data flow of the services is implemented either using a programming language (e.g. Java and C# in our case) or using a BPEL model. In both cases, however, if a service CS3 is deployed in the environment and the Web application wants to take advantage of it, the actual source code needs to be modified. In a future implementation of dynamic composition we want to achieve that the presence or absence of service CS3 can be detected automatically. The Web application combining the services could then dynamically adapt to the different situations at runtime.

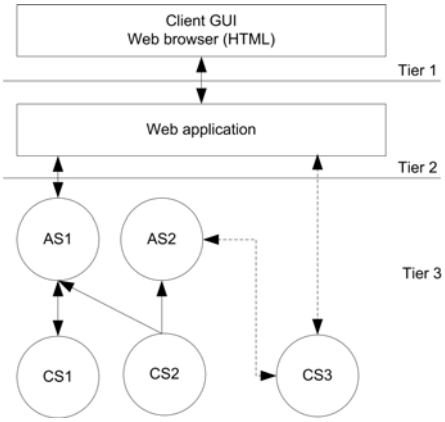

Fig. 2. Web application and services

3 Design and Implementation

3.1 DMC Lab Core Services

Following the architecture and principles discussed in the previous section, we use a bottom-up approach for the realization of our vision. In our case, this means that we first develop core services that run in our environment and then leverage and combine these services to form more powerful services and applications. Currently the core services comprise a presence manager service, an authentication service, a personalization service, a persistent repository service and a service registry. We introduce the functionality of the core services below.

Presence Manager Service

The presence manager offers an interface to determine who is currently in the room. A simple implementation would require an administrator to manually register new persons entering and unregister persons leaving the room. Since this is a cumbersome way of keeping track of the people currently in the room, we use an RFID antenna to automatically detect persons when they enter the room. An RFID antenna detects passive RFID tags when they are in its range and reads the information stored in the tag (depending on the kind of tag used, this can be as little as few bits to several hundred bytes). In our scenario, we assume that every device has such an RFID tag attached to it and that it further has its MAC address stored on the tag. Whenever a person (more precisely a person carrying an RFID-enriched device) enters the room, this is detected by the RFID antenna which in turn notifies the presence manager to update its list of persons. Figure 3 provides a screenshot of the presence manager.

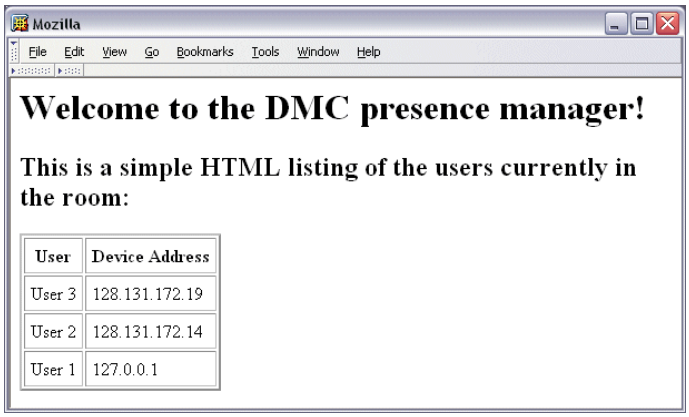

Fig. 3. Presence Manager

Obviously this approach has its limitations. What if a user does not have an RFID tag attached to her device? What if someone enters or leaves the room without her device? In the future, we try to attack the first problem by enhancing the presence detection mechanism with other ways of detecting people entering or leaving the room (e.g., via Bluetooth). We can then combine the results of multiple mechanisms to increase the accuracy of the presence manager. To avoid missing people who enter or leave the room without their device, one option could be to attach the RFID tag to the person rather than the device. Though this will prove difficult in some scenarios, it could work well if all attendees are handed out name tags that have the RFID tag attached. Also tracking of Bluetooth devices could help since people are less likely to enter/leave a room without their cell phone than without their notebook. For our prototype implementation, however, we rely on the RFID technology and the assumption that every device has such an RFID tag attached.

A final design question is what to store on the RFID tags that are attached to the devices. In a first attempt, we stored the IP address of the device together with the user id of the device's owner on the tag. Though this helped us identify who entered the room and what device she is using, this approach is not acceptable in a real-world scenario. First, not all persons entering a room might already have a user id in our environment (e.g., guest, visitors, etc.). Second, we need to support dynamic IP addresses that are not assigned to a specific device. Again a guest visiting our group is a good example of someone obtaining a dynamic IP address for her device. As a consequence, we decided to store the device's MAC address on the RFID tag. Based on the MAC address, we can obtain the currently used IP address. We further store a mapping of MAC address to user id in the presence manager (assuming that users do not share their devices, i.e., the mapping of MAC address to user id does not change). If we detect a device that does not have a user id assigned to it, we automatically redirect the first request from that device to a login/personalization page where the user of this device can specify her identity and preferred settings. Since we assume browser-only clients in the environment, redirecting a request is straight-forward.

Authentication Service

The authentication service is a service that ensures that not only we detected a device entering the room but in addition that the user carrying the device is the same user who the presence manager thinks owns the device. In its simplest form, requests from devices where the user is not yet authenticated are redirected to a login/authentication page. The Web application serving this page would then contact the authentication service to validate the entered user and password information. It depends on the applications and services a user tries to use whether authentication is required or not.

Since sensitive information is transmitted to the authentication service, securing the communication is important. We currently assume that we control the environment strictly enough to keep out intruders from outside the room and that persons in the room are trustworthy. In the future, we plan to replace this assumption by digitally signing the communication with the authentication service using the WS-security recommendation. It remains open how to best manage the required private keys. Potential options include storing them on the user's machine or keeping them in the user's personalization information inside our environment.

Personalization Service

The personalization service stores and offers per-user information. Information is structured in several categories such as general information about a user, information about the user's devices or security settings. The 'general information' section contains the user's name, role and affiliation and is used by other applications (e.g., for displaying a list of user names on the screen or identifying the active user). The device information category contains information about a user's devices such as their processing capabilities or preferences regarding visual presentation (e.g., screen size). The security settings of a user consist of her password or authentication information and potentially a private key for securing communication. Extended user information could comprise how often or how long a user spent in our environment, statistics about the services she used or similar tracking information.

Since security-related information is stored through the personalization service, the communication with the service needs to be secured. Since new users do not yet have a private key or a password, initial communication with the service needs to be secured in another way (e.g., using application-specific rather than user-specific keys).

Persistent Repository Service

In collaborative environments, users meet to work together on some sort of task or action item. Typically, in the process of collaborating, users create, modify and exchange information such as documents (e.g., reviews in the PC meeting scenario) or

notes. Since we assume browser-only client devices, we do not want to store information on the clients since the only way the clients could provide this information to services is via a (manually triggered) HTTP upload. Instead, we use the persistent repository service as a generic store for information created or used in our environment. We currently assume that all information is captured in XML documents and thus use a native XML repository as a backend for the repository service. Via the repository's Web service interface [7, 8], users, services and applications can store, retrieve and query (via XQuery) the XML information in the repository.

Service Registry

The service registry comes in the form of a private UDDI registry that supports publishing and locating services in our environment. We currently use a name-based scheme for the publication and retrieval of services; in the future, a classification-based scheme might replace the simple name-based one. The major difficulty in using a classification-based scheme in our environment is to define the classification itself. Since we try to enable a highly flexible and dynamic environment, it is difficult and might prove ineffective to create a service classification in such an environment. Attaching detailed (maybe machine-readable) descriptions to the service name might be more useful than forcing all services to adhere to a classification.

3.2 Implementation Issues

One of the promises of Web services is that they are platform and language independent. To follow up on this claim, we use a heterogeneous environment in which the core services are implemented in Java or a .NET language (C# in our case) and run on Linux or Windows operating systems. To get us started, we concentrate on relatively powerful machines such as desktop PCs and notebooks. In the future, other platforms and form factors will also be integrated. For the Web service infrastructure, we use the Internet Information Server on the Windows platform and Java Web Service Development Pack (JWSDP) for the Java services on Windows and Linux. The development is done using the Visual Studio.NET IDE for the C# services and a customized Eclipse environment that supports direct deployment of Web services to the JWSDP for the Java services. In the following, we give a short overview of some of the implementation details of the core services.

The presence manager, for example, relies on the RFID client application which is implemented in C# and runs on a Windows machine. The C# application continuously monitors the RFID antenna. Whenever an RFID tag (i.e., device entering or leaving the room) is detected, the application notifies the RFID Web service (implemented in Java and running on a Linux machine). We use WS-I compliant one-way Web service calls to implement notification messages on a Web service platform. The RFID Web service, eventually communicates with the presence manager to keep the list of present persons up to date. As discussed above, other presence detection services could contribute to a more accurate list of users in the room.

The authentication service is a relatively simple Java Web service that currently relies on a local database of user names and passwords. It is designed, however, with a plug-in architecture that supports connecting to other repositories (e.g., a central LDAP directory used for network authentication in our group) to retrieve authentication credentials.

The personalization service is also implemented in Java and already accesses an LDAP directory using JNDI. It defines a set of LDAP classes to store the user-related information outlined above. The service provides an interface to manage and query all the information stored in the LDAP directory. The repository service, eventually, is backed by the eXist native XML database and offers generic operations to add, remove and query its XML content. Also updates (via XUpdate) are supported.

4 Prototype and Future Work

As described in Section 2, the core services are combined in either a static or a dynamic way to form larger services and applications. In our current environment and due to the browser-only clients, applications are realized as Web applications that communicate with the clients via an HTML user interface. As a first step towards the PC meeting scenario and as a test bed for the core services, we implemented a shared task list editor application that statically combines the core services in a Web application.

The purpose of the task list editor application is to enable a group of people in our environment to collaboratively create and update task lists in which tasks are assigned to specific persons and have an attached status indicator (open, in progress, completed). The tasks in the shared task list always reflect the set of users currently in the room. Tasks of users that are not present are either disabled or not shown at all. Further, each user is presented a customized version of the task list in which her tasks appear on the top of the list and are the only ones editable by the user. Eventually, a user can download her tasks and store them locally. Offline modification of the tasks and subsequent synchronization with the task list stored in our environment is currently not supported.

To realize the task list editor scenario, we (statically) combine the core services in a Web application. First the presence manager contributes the list of users currently in the room and thus determines the set of tasks establishing the task list. Based on the information in the user's personalization settings and the device communicating with the application, we provide a customized view of the task list in a layout that matches the user's device-specific settings. Task lists and tasks are stored in the XML database exposed through the repository service. In total, the task list editor Web application takes advantage of the functionality offered by the presence manager, personalization and repository core services.

The service registry acts as an umbrella service that enables finding the other core services used in the scenario. In a next step, we aim at integrating the authentication service to ensure that the persons using the detected devices really are who we expect them to be. The current task list editor is shown in Figure 4.

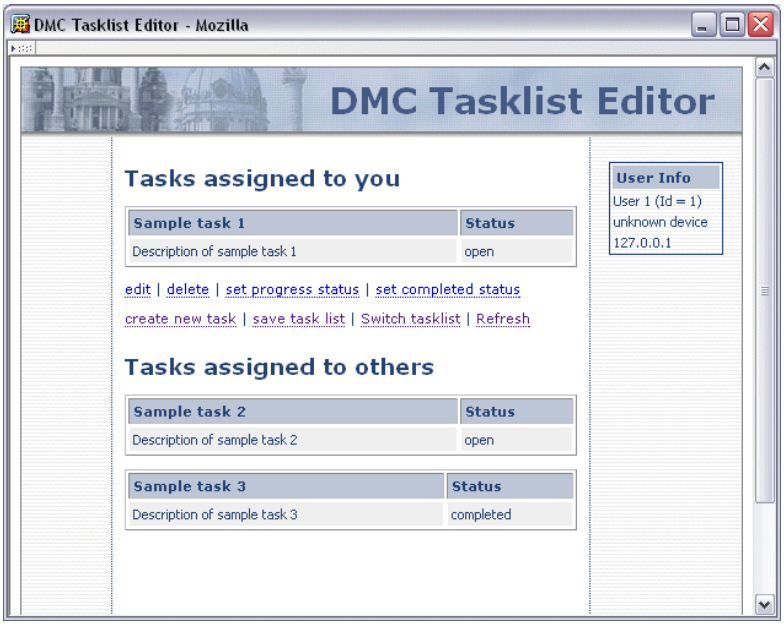

Fig. 4. DMC task list Editor

Although the shared task list editor example combines the core services to form a larger application, it is only a first step towards supporting more complex scenarios like the program committee meeting. It does not, for example, involve BPEL processes that combine other services; neither are advanced security or concurrency issues of importance currently.

To better support complex and ad-hoc processes, we are now implementing a trust service that manages authorization settings and adjust depending on a user's actions and behavior. We also develop workflows that are represented as BPEL processes and thus are reusable and composable. Such processes are deployed to BPEL engines such as IBM's BPWS4J or Microsoft's BizTalk and are themselves exposed as services. In situations such as voting scenarios, digitally signed messages might be interesting to ensure that everybody voted only once or to be able to determine the sender of a message. Consequently, we are currently integrating digital signatures based on the Web service security stack.

In the future, additional and improved services, technologies and scenarios can be envisioned. For example, presence detection does not need to exclusively rely on RFID technology but could be a combination of several detection approaches as outlined in Section 3.1. We are also experimenting with scenarios where multiple RFID antennas at strategic positions in a room can support richer scenarios (e.g., to automatically detect who is presenting, etc.). New technologies such as Microsoft's upcoming Indigo communication infrastructure can also be a powerful extension of our environment.

We currently manually code the communication between the (HTML-based) user interface and the collaborative processes. In the future, the connections between the GUI and the business processes or Web applications should be at least semi-automatically created based on XForms similar to the approach outlined in [13]. If we further loosen the requirements for our scenario and assume that we deal with more powerful client devices that not only can run a Web browser but provide services on their own, we introduce a new level of complexity in terms of service registration and discovery and (dynamic) service composition. Also new challenges in the areas of consistency, replication and synchronization arise in a setting where services reside on client devices and can be used detached from our environment.

A final direction of future work deals with the integration of audio capabilities into our environment. Support for VoiceXML, for example, could provide for spoken instructions.

5 Conclusion

Recent advances in Web services show the potential in designing flexible and loosely coupled systems utilizing service composition techniques and languages. Presence-aware technologies such as RFID allow for low-cost technologies to help in localizing objects carrying RFID tags. In this paper we have shown how both technologies can be combined to provide a powerful presence-aware infrastructure. We discussed design and implementation issues of our first prototype and presented our suggestions for improvements. We can conclude that a combination of Web services technologies on the one hand with presence-aware technologies on the other hand seem very promising for environments aiming at supporting same room/same time collaborative scenarios.

Acknowledgements

This research was supported in part by the ERASMUS program of the European Union.

6 References

[1] Universal Description, Discovery and Integration: UDDI Technical White paper. http://www.uddi.org/pubs/Iru_UDDI_Technical_White_Paper.pdf. 2000
[2] W3C. Web services Architecture W3C Working Draft 8 August 2003. http://www.w3.org/TR/2003/WD-ws-arch-20030808/wsa.pdf. 2003
[3] Fabio Casati, Ming-Chien Shan. Dynamic and adaptive composition of e-services. Software Technology Lab, Hewlett-Packard Laboratories. 2001
[4] UDDI Version 2.03 Data Structure Reference. http://uddi.org/pubs/DataStructure_v2.htm. 2002
[5] UDDI Version 3.0.1 http://uddi.org/pubs/uddi_v3.htm. 2003
[6] W3C. SOAP Version 1.2 Part 0: Primer. http://www.w3.org/TR/2003/REC-soap12-part0-20030624/. 2003
[7] W3C. WSDL, Web services Description Language.

http://www.w3.org/TR/2002/WD-wsdl12-20020709/. 2002
[8] Using WSDL in a UDDI registry, Version 2.0 http://www.oasis-open.org/committees/uddi-spec/doc/tn/uddi-spec-tc-tn-wsdl-v200-20030627.htm. 2003
[9] W3C. XML Extensible Markup Language. http://www.w3c.org/XML. 2000
[10] BPEL4WS (2003). Business Process Execution Language for Web services specification, http://www-106.ibm.com/developerworks/library/ws-bpel/
[11] W3C. XQuery 1.0: An XML Query Language. http://www.w3.org/TR/2003/WD-xquery-20031112/. 2003
[12] M. Jazayeri (2003). Pervasive software services for mobile ad-hoc teams. Proceedings of the 1st International Workshop on Ubiquitous and Mobile collaboration Systems (UMICS 2003), co-located with CAiSE 2003, Velden, Austria.
[13] S. Perkles (2003). Server-side XForms Processing - A Browser-Independent Implementation of the Next Generation Web Forms. Master Thesis, Vienna University of Technology, 2003, Austria.