TWSAPI: A Generic Teamwork Services Application Programming Interface

Engin Kirda, Gerald Reif, Harald Gall and Pascal Fenkam

Technical University of Vienna, Distributed Systems Group
Argentinierstrasse 8/184-1, A-1040 Vienna, Austria
{E.Kirda, G.Reif, H.Gall, P.Fenkam}@infosys.tuwien.ac.at

Abstract

One of the problems faced by large, global organizations and enterprises is to effectively enable their employees to collaborate across locations. People need collaborative work support while they are on the move and have to share business documents and know-how. Although much work has been done in the area of Computer Supported Collaborative Work (CSCW) to date, supporting mobility is only recently receiving attention. Hence, most of the existing approaches do not deal with emerging mobile teamwork requirements such as locating business documents and expertise through distributed searches, advanced subscription and notification, community building, and mobile information sharing and access. Furthermore, existing applications and approaches are usually difficult to customize to business-specific processes and requirements. The MObile Teamwork Infrastructure for Organizations Networking (MOTION)[1] *system addresses these requirements and provides a generic teamwork services Application Programming Interface (API), TWSAPI, that can be used to build organization-specific collaborative applications. In this paper, we give an overview of the MOTION TWSAPI and illustrate its usage in building an application that provides document review support.*

Keywords: MOTION, Mobile Teamworking, Information and Communication Technology Architecture, Mobile Computing

1 Introduction

Many large, global organizations are distributed across sites in different countries and continents. Hence, they have to provide an effective IT infrastructure to enable their employees to share information and collaborate. The difficulty of the problem is that the nature of working is constantly changing. With the major advances in mobile computing in the last decade, employees are now frequently on the move with their notebooks, Personal Digital Assistants (PDAs) and mobile phones. Being on the move, hence, has become *part* of working and some have referred this trend as *nomadic working* (e.g., [10]).

Varnum in [12], for example, discusses how PDA-services were deployed at Ford. The problem was that managers and company leaders that were higher up in the company hierarchy did not have any time to get information from the company intranet. These people were always busy and only had time between meetings. They preferred to get their emails in paper form and had time for correspondence in cars, during flights, etc. It was observed that PDAs had found a high acceptance by these people. The IT department decided to utilize the wide usage of PDAs (i.e., Palms in this case) and developed a system with which Web sites in the intranet can be downloaded to these devices.

Employees not only need *access to information* while on the move, but also need *collaborative support* so that they can share business documents and know-how. Although much work has been done in the area of Computer Supported Collaborative Work (CSCW) to date, supporting mobility is only recently receiving attention. Hence, most of the existing approaches do not deal with emerging mobile teamwork requirements such as locating business documents and expertise through distributed searches, advanced subscription and notification, community building, and mobile information sharing and access. Existing applications and approaches, unfortunately, are usually difficult to customize to business-specific processes and requirements.

For the last two years, we have been working on a mobile teamwork system in the MObile Teamwork Infrastructure for Organizations Networking (MOTION) project that addresses these requirements and provides a generic team-

[1]This project is supported by the European Commission in the Framework of the IST Programme, Key Action II on New Methods of Work and eCommerce. Project number: IST-1999-11400 MOTION (MObile Teamwork Infrastructure for Organizations Networking)

work services Application Programming Interface (API): TWSAPI. The API can be used to build organization-specific collaborative applications. The system was designed on the distinct requirements of two global, well-known enterprises.

In this paper, we give an overview of the MOTION TWSAPI and illustrate the usage of the services it provides in building an application that supports document reviews.

The paper is structured as follows: The next section gives a brief introduction to the two industrial case studies in the MOTION project. Section 3 lists the general requirements for a system that needs to support mobile teamwork. Section 4 describes the MOTION Teamwork Services API (TWSAPI) and describes the usage of the API in building a collaborative application in one of the case studies. Section 5 presents related work and Section 6 concludes the paper.

2 Case study organizations

In this section, we give a brief overview of the industrial case studies in the MOTION project and discuss the mobile teamwork-related challenges.

2.1 Case study: Mobile phone design

The first case study is a global company involved in the market of telecommunication equipment and systems. The company is a well-known manufacturer and designer of mobile phones. It has branches distributed in several continents.

The company has highly specialized domain experts that are involved in many research projects and that travel frequently between the research centers. Although much know-how exists in the organization, because of the distributed nature of the company, developers are sometimes unaware of technical problems that already have been solved and are unable to locate and communicate with experts that are in other locations. Support, hence, is needed to arrange face-to-face meetings (e.g. for reviewing a document) and synchronous communication sessions (e.g., chat, voice- or video-streaming).

After looking at existing collaborative systems in the market, the company was not able to find a product that would increase the availability and working efficiency of its employees by providing support for mobile working, knowledge sharing and collaboration.

2.2 Case study: Household Appliances

The second industry case study is a global, well-known producer of white goods. The company's manufacturing experts travel around the world. In many cases, the information the experts need is not in one location, but is distributed across repositories in factories and sites. When setting up a new factory in a third world country, for example, it is useful for the experts to find out information about similar situations and settings in other countries. Because of the lack of an appropriate information infrastructure, this information is sometimes lost and is not accessible.

Hence, the main problem for the company is to support the experts in querying distributed knowledge repositories to increase productivity, decrease the costs and time needed to complete certain tasks.

3 General services for mobile teamwork

After analyzing the requirements of both companies, we identified the following general services for providing mobile teamwork support.

- **Distributed searches**: Information in companies are distributed across computers. A typical employee may have documents on his computer that are interesting for other employees, but that cannot be shared because they are not stored in a traditional central repository. Support is needed, hence, in specifying and issuing distributed queries across personal computers and computing devices (e.g., PDAs) to locate and share information.

- **Meta data**: Many documents exist in companies that do not really belong to well-defined tasks. Examples of such documents are pictures of a factory an expert may have taken, notes an employee has written down in a conference and a sketch for the solution of a design problem. In order to share and locate this information effectively, mechanisms are needed that allow the description of the contents of documents and business artifacts. Flexible, customizable meta data support is needed for companies that allow employees to define keywords and short descriptions of their documents.

- **Communities**: Mechanisms are needed that allow employees to build virtual communities that enable them to collaborate regardless of where they are are and what computing device they are using. The concept of communities has been increasingly gaining popularity in collaborative products and systems. Examples of communities are: Mobile phone design, system administrators, software engineering research, etc. Employees should be able to set up communities themselves to share information.

- **Subscription and notification**: Employees often depend on work done by others. In many cases, they have to wait for someone else to finish before they can continue. When writing a research paper, for example, each author might write a part of the paper and send it to the chief author who then compiles and edits the full paper. Subscription and notification support was one of the most important requirements we identified in the global companies. Employees wished to be notified on information they were interested in. Hence, subscription mechanisms are needed that enable them to subscribe to and get notified of events. In the paper writing example, the chief author would receive notifications when each co-author has finished writing her part. Note that employees were mainly interested in short notifications on the availability of information and *not* the delivery of the information itself.

- **Mobility and device-independence support**: A wide range of fixed and mobile computing devices such as desktop computers, notebooks, PDAs or WAP-enabled mobile phones have to be supported. Hence, the user should only be limited by the capabilities of the device she is using. Note that mobility support targets both user and device mobility. In a case when the user does not have a computing device with herself, she should be able to use the existing Internet infrastructure such as the World Wide Web (WWW).
- **Access control**: Access restrictions are needed when sharing information. Hence, support for user names, passwords, encryption and certificates are needed.

4 TWSAPI: The Teamwork Services API

In this section, we give a brief overview of the MOTION system architecture and present the MOTION Teamwork services API.

4.1 Overview of the MOTION architecture

The MOTION system has a layered architecture consisting of three layers. The bottom layer is the communication middleware offering basic services such as publish/subscribe mechanisms (i.e., event-based system support), peer-to-peer file sharing functionality and distributed search propagation. In the current prototype we have implemented, the communication middleware infrastructure is provided by Peerware [9]. The bottom layer, however, can be replaced by any other components that can provide distributed search and publish/subscribe support such as JTella [7] or JEDI [2].

We refer to every computing device that is connected to the MOTION system as a *peer*. Some peers in the system host services and some only act as clients. Any computing device that is able to run the MOTION libraries can act as both, service host and client. A typical MOTION configuration, thus, consists of desktop computers, notebooks, and PDAs that can host services and also act as clients. Clients such as Web browsers or WAP enabled phones do not host services, but access them remotely.

Each peer that has the technical capability hosts a repository. Artifacts (i.e., documents and files) that the user wishes to share with others in communities are stored in this repository along with the corresponding XML metadata (i.e., *profile* in the MOTION terminology).

The repositories on peers also store community and user profile information that are replicated across peers and are synchronized using events. From the user perspective, profiles are used to support queries based on descriptions of *resources* (i.e., communities, users and artifacts). From the system point of view, these profiles are also used to store system-relevant information such as user names and last modification dates.

The middle layer in the architecture, the Teamwork Services (TWS) layer, is built on top of the communication middleware. It is responsible for the integration of the main components of the system such as the repository and the access control and management component. Furthermore, it provides an Application Programming Interface (API) to the following generic services:

- Storing and retrieving artifacts in the local repository and from remote repositories on other peers
- Managing *resources* (artifacts, users, and communities) and their profile information.
- Creation and management of communities
- Sharing of artifacts with other users within communities
- Subscribing to specific events in the MOTION system (e.g., artifact insertion, artifact deletion, user is online, etc.) using the XML Query Language (XQL).
- Sending and receiving messages from other users
- Receiving system messages and notifications (e.g., subscriptions).
- Distributed searching for artifacts, users and communities based on their profiles (using XQL)

Using the TWSAPI and the generic service functionality it provides, the application programmer can build Collaborative Business Specific Services (CBSS) that are custom-tailored to the requirements of an organization. In Section 4.3, we present one such CBBS that is currently being built in the first industrial case study company.

The top layer in the architecture is the presentation layer. The different user interfaces the MOTION system is able to support are in this layer. A user interface for MOTION is built using the TWSAPI. Because of mobility requirements, a typical configuration will have different interfaces for desktop/notebook computers, PDAs, WAP-enabled mobile phones and Web browsers. In the prototype of the system, we have implemented a Java user interface for notebooks and PCs and are currently working on PDA and Web interfaces.

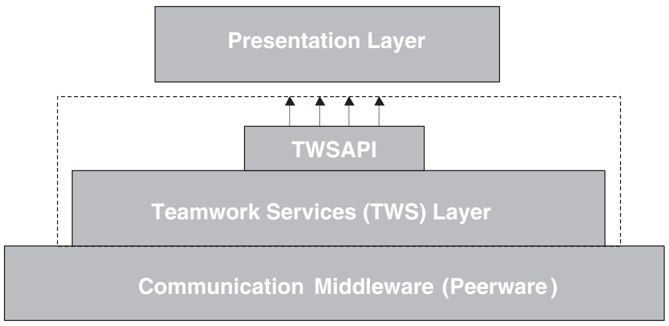
Presentation Layer
TWSAPI
Teamwork Services (TWS) Layer
Communication Middleware (Peerware)

Figure 1. The layered architecture of the MOTION system

Figure 1 depicts the layered architecture of the MOTION system. The TWSAPI provides access to the functionality provided by the Teamwork Services layer.

4.2 Main components of the TWSAPI

We have implemented the prototype of the TWS layer and the TWSAPI in Java. The system libraries are small enough to run on most devices that support the Java Development Kit (JDK) Version 1.1. We have tested the code under Windows, Linux and on Compaq iPAQ PDAs running Familiar Linux.

The TWSAPI consists of six main components: The *MOTIONManager*, *ArtifactManager*, *SubscriptionManager*, *MessageManager*, *CommunityManager* and *UserManager*.

Figure 2 depicts the conceptual class diagram of the TWSAPI.

When a user starts a MOTION session on his local peer, the MotionManager is the first class of the TWSAPI that is instantiated. Once the user has an instance of the MotionManager, he can work offline with the resources on his local peer. The MotionManager, for example, returns an instance of the class ArtifactManager that is responsible for accessing the local artifacts or issuing queries on them. When the user wishes to work in communities and access resources hosted on other MOTION peers, he has to connect to the MOTION system and authenticate himself. He can connect to any peer on the system. In a typical configuration, a number of peers are always online and act as servers (i.e., backbone computers). Once the user is connected, he and his shared resources are visible to other users and can be searched globally.

The ArtifactManager provides methods for inserting, deleting, copying and moving MOTION artifacts into the local repository or to the shared repository of a community (i.e., *community cabinet*)[2]. If a user *johnWayne*, for example, wishes to share a research paper in the community *Mobile phones*, he would first insert the document into the repository of the peer he is using (with relevant artifact profile information) and then he would tag the artifact as belonging to the community. In the background, the client application that he is using would use the methods provided by the ArtifactManager to achieve this task.

The CommunityManager is responsible for providing methods directly related to community tasks such as creating a new community, removing existing ones and defining subcommunities in communities. The CommunityManager also provides general utility methods such as searching community profiles and retrieving a list of communities the user is a member of.

The SubscriptionManager component can be used to create, retrieve and manage user subscriptions. It can be used, for example, to define a new subscription to documents that are created by the user *johnWayne*. MOTION subscriptions are based on the Java Event Model. When a user creates a subscription, the TWSAPI defines an event filter and an event listener (i.e., SubscriptionListener interface in the API). Whenever an event occurs that matches the event filter, the corresponding event listener provided by the application programmer is invoked. The event listener (i.e., the implementation of the SubscriptionListener interface) is then responsible for reacting to the event (e.g. show a message to the user). Subscription criteria are provided in XQL. The application programmer, hence, has to provide easy mechanisms for users to define subscriptions. In our user interface prototype, for example, we use graphical tables that generate XQL criteria that are then passed to the TWSAPI.

The MessageManager component is used to send user-to-user messages within the MOTION system. The MOTION message handling is also based on the Java Event Model. The application programmer using the TWSAPI, hence, provides implementations of the MessageListener interface that are invoked whenever the user receives messages. (i.e., from the system or other users).

The UserManager component is used to create, delete and manage users. The access rights of users are managed by DUMAS [4] that is a generic user management compo-

[2]The community cabinet allows users to persistently store information on other peers so that the information is visible to other community members even when they are offline

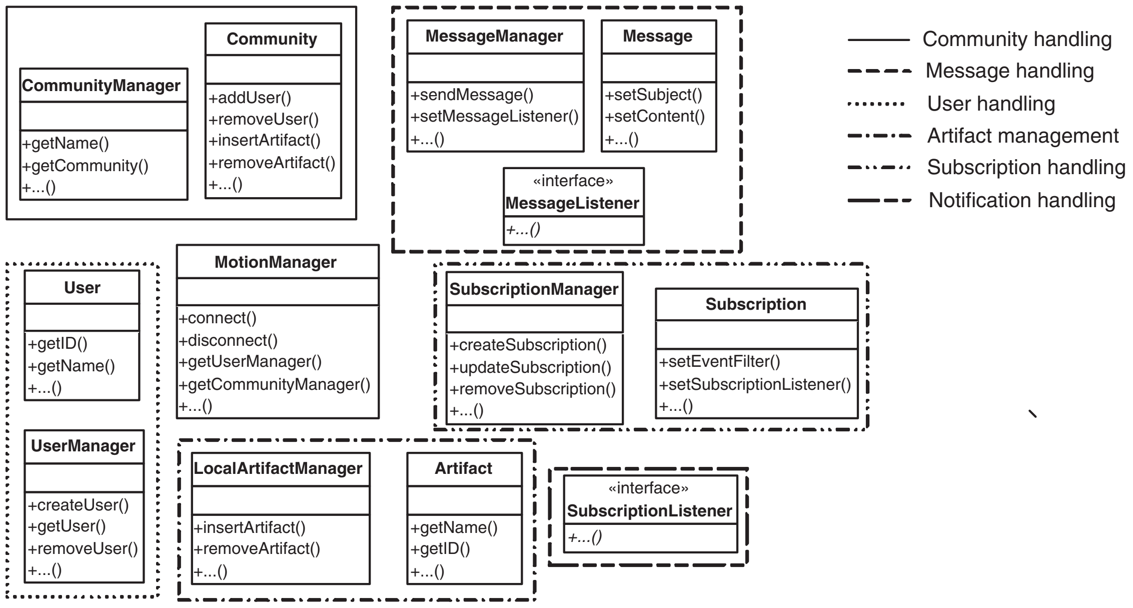

Figure 2. The TWSAPI conceptual class diagram

nent. The UserManager in the TWSAPI acts as a wrapper to the functionality provided by DUMAS.

Artifacts, users and communities in the MOTION system are represented by classes that provide methods to access and modify the profile information.

4.3 TWSAPI usage example: Document reviewing

In this section, we discuss a typical business-specific mobile working scenario that is currently being implemented with the generic MOTION TWSAPI for the first case study company. The company needs teamwork support for document reviews (see Section 2.1).

Dr. X is sitting in Vienna and finishes a document he has been working on for the last few weeks. All he needs now is the approval of his project manager, Dr. M, to start the review process for the document.

He creates a new community for the review and places the document into the community cabinet. The review community is a subcommunity of the project community. Template documents for the review such as a defect list are automatically generated when the review community is created. After he has inserted the document into the review community, the project manager, who is automatically member of this subcommunity, is informed with a message that Dr. X has finished editing the document. Dr. M downloads the document on her notebook and checks whether the document is ready for review. She accepts it and delegates the task of preparing the review meeting to Dr. X.

Dr X. is informed with a message about his new task. First, he has to find reviewers that are experts in the document's domain. Therefore, he searches for experts in the MOTION system (i.e., in the user profiles). The search result contains 10 experts whose user profiles match the query criteria. He selects seven of the experts and asks them to review the document. The experts are added to the review community and a message is sent to all of them. In addition, access rights are set so that they have the rights to access the necessary documents. The experts can then decide whether they can participate in the review meeting or not. In case a reviewer cannot physically attend a review meeting, she can offer to work as a *distant reviewer*. Distant reviewers follow the review via messaging. The invitation for the review includes a few review dates the reviewers can select from. The message is then sent back to Dr. X.

One of the reviewers downloads the document and starts reading it on his PDA. He finds that the document is in good shape and does not have major problems. He adds some minor remarks to the defect list.

The following morning, Dr. M checks the review community and sees that 5 of the 7 requested reviewers have accepted to participate in the review meeting. One participant has declined (she is away and has responded with her WAP-enabled mobile phone) and one is participating as a distant reviewer. All of the participants have already reviewed the document and have added their comments to the defect list. Dr. M also checks the possible dates of the review and chooses one that is suitable for all reviewers. A message is sent to all community members to inform them of the date and location of the review. Dr. X is now responsible for setting up an agenda for the meeting and to place it into the review community.

Finally, the review meeting day has arrived. Dr. M, as the manager of the project, checks whether all participants are present (either physically or connected to the MOTION system). The distant reviewers are able to follow the review by viewing the agenda and the common defect list in the community cabinet. They can send their comments to the review community via MOTION messages.

The review is ending and as a final task Dr. M sums up the results of the meeting and repeats the action points that she has included in the meeting minutes. Distant reviewers are to check the meeting minutes in the following week and include any comments they may have. Dr. X needs to do some small changes and include a couple of related issues to the document.

5 Related Work

Although much has been written to date on Computer Supported Collaborative Work (CSCW), the majority of these systems are not customizable and are not concerned with device and user mobility.

StudySpace [11] is one of the first projects to tackle mobility in collaborative applications. It does not address subscription, distributed searching and community support issues, though.

Several authors have advocated the need to develop groupware systems that support user mobility. DACIA[6] is a system that provides mechanisms for building groupware applications that adapt to available resources. The system, however, does not support higher-level services that we address. In MOST[1, 3], the functionality provided by the group coordination module is comparable to the community support provided by MOTION. The goal of MOTION, however, is not to provide a single collaborative application, but a generic framework and platform for the development of business-specific applications. Further, MOST does not support emerging mobile teamwork service requirements that we have discussed.

Sync[8] is a Java-based framework for developing collaborative applications for wireless mobile systems that is based on object-oriented replication and offers high-level synchronization-aware classes based on existing Java classes. MOTION differs from Sync because it provides a generic API to higher-level teamwork services.

Several commercially available collaborative systems (e.g.,[5]) have started to support some form of mobility (e.g., WAP access), but they are usually not customizable. Further, they were reported by our case-study companies as having insufficient teamwork support and not covering the services we address in this paper.

6 Conclusion

In this paper, we gave an overview of the MOTION Teamwork Services API (TWSAPI) and discussed the usage of the services it provides in building an application that supports document reviews.

In our two case studies, we observed a growing need for generic mobile teamwork frameworks and Application Programming Interfaces (APIs) in large, distributed organizations.

The MOTION Teamork Services prototype is currently being evaluated in the case study companies.

References

[1] K. Cheverst, G. Blair, N. Davies, and A. Friday. The support of mobile-awareness in collaborative groupware. *Personal Technologies*, 3(1-2):33 42, 1999.

[2] G. Cugola, E. D. Nitto, and A. Fuggetta. The JEDI Event-Based Infrastructure and its Application to the Development of the OPSS WFMS. *Transaction of Software Engineering (TSE)*, 27(9), September 2001.

[3] N. Davies, G. Blair, K. Cheverst, and A. Friday. Supporting collaborative applications in a heterogeneous mobile environment. In *Computer Communications Special Issue on Mobile Computing, Internal report number MPG-94-18.*, 1996.

[4] P. C. Fenkam. Dynamic user management system for web sites. Master's thesis, Graz University of Technology and Vienna University of Technology, September 2000. Available from http://www.ist.tu-graz.ac.at/publications.

[5] P. Inc. Pragmatyxs, http://www.pragmatyxs.com, 2002.

[6] R. Litiu and A. Prakash. Developing adaptive groupware applications using a mobile component framework. In *ACM 2000 Conference on Computer Supported Cooperative Work. ACM, New York, NY, USA*, page 107 116. ACM Press, 2000.

[7] K. McCrary. Jtella homepage, http://www.kenmccrary.com/jtella/, 2002.

[8] J. Munson and P. Dewan. Sync: a java framework for mobile collaborative applications. *IEEE Computer*, 30(6):59 66, June 1997.

[9] G. P. Picco and G. Cugola. PeerWare: Core Middleware Support Peer-To-Peer and Mobile Systems. Technical report, Dipartimento di Electronica e Informazione, Politecnico di Milano, 2001.

[10] J. Schiller. *Mobile Communications*. Addison-Wesley, Reading, Mass. and London.

[11] J. L. Schnase, E. L. Cunnius, and S. B. Dowton. The studyspace project: Collaborative hypermedia in nomadic computing environments. *Communications of the ACM*, 38(8):72–3, August 1995.

[12] K. Varnum. Information @ your fingertips: porting library services to the pda. *Online*, 24(5):14 17, September - October 2000.