# Pervasive software services for mobile ad-hoc teams

Mehdi Jazayeri
Technische Universität Wien
jazayeri@tuwien.ac.at

**Abstract**
Mobile, ad hoc teams are an increasingly common form of collaboration. Workers in such teams often are involved in several highly information-intensive projects. They work with people in several organizations, on short-term, goal-oriented tasks. Most of today's information technology-based tools have been developed to support traditional, static, project-oriented teams. The support of mobile, ad-hoc collaboration offers many challenges to tool developers. The paper considers the application of pervasive computing services to the problem. The paper is speculative, but, I hope, stimulating.

**Keywords**: software architecture, ad-hoc teams, mobility, cooperation services, teamwork services

## 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. The worker has to regularly sift through incoming and past information to determine the information relevant to his or her current context. How effectively the worker is able to perform his or her task depends on how well he or she can remember the relevant information and documents, remember their relationships to one another, and maintain access to them while on the move. As tasks become more complex, the relationships among documents become even more complex. To make matters worse, knowledge workers are typically involved in several tasks at the same time, some short-lived and others long-term. Keeping track of all tasks and their dependencies has become a formidable endeavor. The emergence of the "pervasive computing" environment, also referred to as "ambient intelligence", promises the availability of essentially unlimited computing and communication capability regardless of location. How these capabilities can be exploited is a matter of current speculation and limited by our imagination. We envision that task support for knowledge workers can be pushed into the intelligent ambient in each working environment. We will analyze a typical team project—paper selection by a program committee for a conference—to illustrate typical requirements, show how this project may be sup-

ported by the MOTION team support environment, and explore possibilities for embedding this support in the pervasive computing environment. A fundamental restructuring of system and application software architecture is necessary to achieve this kind of support.

## 2. A typical scenario of ad hoc team cooperation

*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 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, much 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.

### 2.1 Pre-meeting phase

In the pre-meeting phase, the committee drafts a call for papers (CFP) and publishes it in appropriate venues, soliciting papers for the conference. Today, this is done mostly on a conference Website and through (repeated) electronic mail postings. The CFP contains instructions on how to submit a paper, in what format, any length restrictions, and special requirements such as copyright releases. Many conferences also have restrictions such as rules against simultaneous submissions to a different conference or journal. Committee members also try to informally spread the word about the conference and solicit good papers. This practice is based on the assumption that the committee members, being experts, know the other experts in the field, who are likely to be performing work of interest to the conference.

Once the CFP is published, potential authors around the world consider the conference and some prepare and submit papers according to the instructions in the CFP. Today (2003), most submissions are done electronically through a Website; physical paper submission through the post office is not supported. For perspective, it is interesting to note that in 1999, very few conferences supported electronic submission.

The submitted papers are stored in a repository. There are many conference paper submission systems that support this process.

Once the deadline for paper submissions has expired, the PC Chair assigns papers to reviewers from the PC. This itself is an intricate process. First, the Chair collects an expertise and preference list from PC members. The criteria for assigning papers for review are usually based on this information but also have to balance the load among reviewers and also avoid potential conflicts of interest. For example, a paper should not be assigned for review to the author's close friends, colleagues, or competitors. Depending on the size of the conference, each paper may be assigned three or more reviewers, and each reviewer may be assigned up to twenty papers to review. Sometimes, PC members may ask another person to review the paper. These "external" reviewers become virtual members of the PC but are not present at the PC meeting.

The reviewers have a deadline for completing their reviews. This means that they must read the paper, prepare a detailed review with explicit comments for the authors and for the committee, together with a recommendation as to whether to accept or reject the paper. The reviews must be submitted a week or two before the meeting. Today, these reviews are submitted electronically and stored in a repository. Depending on the policies chosen by the PC Chair, the reviews for papers may or may not be visible to other reviewers of the same paper. If the reviews are visible, the reviewers have a chance to discuss their possibly divergent opinions and perhaps reach an agreement about the status of the paper before the meeting. Typically, the reviews are not visible to those with a conflict of interest with the paper.

The Chair reviews the reviews to determine if there are any problem cases. An example problem case is if the reviewers indicate that they do not have sufficient expertise to be confident in their review. Another example is when there is real difference of opinion among the reviewers. In such cases, the Chair may decide to get an additional review on the paper, either from a member of the PC or externally. On the basis of the reviews, the Chair determines what papers should be discussed at the program committee meeting. Sometimes the choice of papers to be discussed is communicated to the PC beforehand and other times only at the beginning of the meeting.

### 2.2 Meeting phase

The reviewing work is done before the PC meeting, but the meeting is where final decisions about the disposition of the papers are taken. The meeting is thus the culmination of the PC's work. Since the PC members are from different organizations and from different parts of the world, most members have to travel to the meeting place. Sometimes, meetings are even held at airport lounges to ease the travel burden for everyone. The point is that most members, possibly all, are away from their home environment. Sometimes, Internet access is available and other times not.

The goal of the meeting is to discuss the papers and their reviews and to decide which papers should be accepted for presentation at the conference. Typically, the papers with uniformly poor reviews are not discussed, based on the assumption that if all reviewers disliked the paper, no one is going to argue that the paper should be accepted. On the other hand, papers with uniformly positive reviews are discussed so that all PC members are informed of the top papers in the conference. This discussion is not strictly aimed at reaching the primary goal of the meeting but is intended for community and awareness building. Most of the meeting is devoted to discussing papers with mixed reviews.

The meeting is usually supported by some kind of management software. Typically, the paper to be discussed is displayed on a screen, along with the reviewer names. These days, all members have laptops with stored reviews of all papers. The Chair decides the order in which papers are to be discussed and the order in which reviewers start the discussion. Sometimes, papers with positive reviews are discussed first, sometimes those with negative reviews, and other times in some mixed order. The policy tries to optimize the efficiency of the process but also should take into account group psychology. Those members who have conflicts of interest with the paper being discussed usually leave the room and do not take part in the discussion. A trivial example is a paper's author who must not be present when his or her paper is being considered. Many conferences have guidelines for handling papers authored by program committee members to avoid conflicts of interest. Sometimes, PC member papers are subjected to more reviews or higher standards.

The PC Chair moderates the meeting. The reviewers present their opinions about each paper and argue the points in favor and against the paper. The PC Chair tries to determine from the arguments whether the paper should be accepted or rejected. Sometimes there are middle-ground decisions such as accepting the paper for a poster session rather than a regular paper. Sometimes there are categories of papers such as research papers versus industrial papers versus experience reports or case studies. Sometimes explicitly different standards are applied to the various categories and sometimes the standards are implicit. The goal is to reach a consensus decision about each paper. Sometimes a consensus is not possible and the discussion is about the risk of accepting a possibly bad paper versus the risk of rejecting a possibly good paper. Eventually, the Chair has to finalize a decision.

At the conclusion of the meeting, a set of papers has been selected for the conference, possibly categorized in different classes.

### 2.3 Post-meeting phase

The goal of the next phase of the process is the preparation of the Proceedings for the conference. Before that can be done, the Chair must communicate with the authors of the papers about the status of their papers. First, the reviewers are asked to update their reviews, if necessary, based on the discussions at the PC meeting. It is possible that some arguments have revealed new insights that should be communicated with the authors, either to explain the reasons for the rejection of the paper or to help the author improve the paper. Once the reviews have been updated, the PC chair sends a notification letter to the authors announcing acceptance or rejection of the paper. The letter is accompanied by reviews that explain the decision to the author and sometimes contain suggestions for improving the paper. The authors of the accepted papers are given a deadline for submitting "camera-ready" versions of their papers, appropriately improved, in a prescribed format, together with a signed copyright release form.

The camera-ready papers are collected together by the PC Chair to form the core of the Conference Proceedings. They are sent to the publisher who publishes the Proceedings and delivers them to the conference site.

## 3. Technological support for PC meetings

It is interesting to examine the development of technology and how it has been applied to the PC meeting problem. Before 1990, most paper submissions were done on paper and through the post office. In early to mid 1990s, reviewers generally emailed their reviews of papers to the PC Chair. In 1997, I chaired the PC of European Software Engineering Conference and the ACM Symposium on the Foundations of Software Engineering (ESEC/FSE). We had physical submissions of papers through the regular post office. We sent by courier mail copies of papers to be reviewed. Reviewers emailed their reviews. We wrote Perl scripts to process the reviews and produce reviewer booklets for the PC Meeting. The process was primarily manual.

In 2000, I co-chaired the PC Meeting of the International Conference on Software Engineering (ICSE). We encouraged electronic submissions. We received two papers (from the same person) in paper form. The rest of the 350+ papers were submitted electronically. We experienced some surmountable problems in postscript/pdf compatibility. We used the Cyberchair paper management system (www.cyberchair.org/). Each reviewer had a user/password to access the system, download the papers assigned to him or her, and submit reviews on the papers. After submitting a review, the reviewer could see the other reviews and could start discussing the differences with the other reviewers. This discussion was through email, outside the system. The PC chairs were "expected" to be copied on the discussion emails. The electronic submissions eliminated the cost of courier mailing entirely, both for the authors and for the conference. The cost was replaced by the additional task of printing of papers by reviewers.

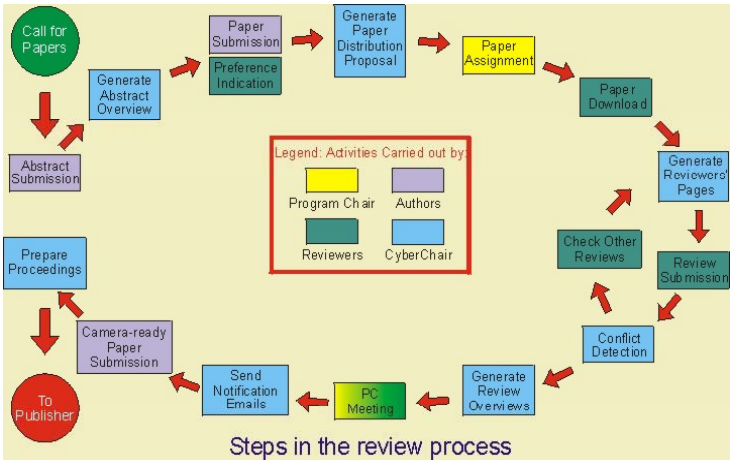
To prepare for the discussions of the committee meeting, we produced a huge booklet containing all the reviews of the papers, individualized for each of the 50 members of the committee. In 2003, when I attended the PC meeting of ESEC/FSE as a member, the Chair distributed (electronically) all the relevant reviews beforehand and each member carried a laptop to the meeting with copies of all the papers to be discussed and all the reviews. There were hardly any physical papers at the meeting. On the other hand, in 2000, when members left the meeting, they left the booklets at the meeting, respecting their confidentiality. In 2003, we received an email from the PC Chair (after the meeting) to remove the reviews from our laptops.

Certainly the nature of program committee meetings has changed with increasing use of technology. The changes in process have also affected the authors, reviewers and publishers. Today, most conferences are supported by some management software that provide, to different degrees, paper submission, reviewer registration (with lists of competences), paper assignment to reviewers, review submission, discussion forums, ordering of papers for discussion, camera-ready submissions, and Proceedings production. There are also systems that provide other services such as marketing of the conference via emailing to appropriate Websites. A major change that has enabled a completely new approach to the meeting is that almost everyone has a laptop computer. The use of the individual computers enables a much more interactive and

efficient discussion at the meeting (at least by those members who are not playing solitaire or reading email!).

It is interesting to consider the innovations that will be possible in this scenario with the application of pervasive computing. Pervasive computing has the potential to augment human performance by providing ubiquitous access to information, processing, and services in a truly distributed fashion. The program committee-meeting scenario is a prototypical example for investigating possible applications of pervasive computing because it is a typical teamwork scenario, requiring support for highly flexible processes. We will first consider our meeting scenario in the context of an existing team-support software before taking up the issue of pervasive computing.

The figure below is an overview of the activities of a PC from the point of view of Cyberchair, reproduced from the Cyberchair Website. The Website also contains more precise definitions of terms author, reviewer, and so on, and links to related papers. A paper by Oscar Nierstrasz [Nierstrasz] proposes a pattern language for identifying the "champion" for a paper in a PC Meeting, the person who will argue that the purpose should be accepted. The pattern is used in many conference meetings.




## 4. Implementing teamwork in MOTION

In the last few years, the EU project MOTION (www.motion.softeco.it) developed a teamwork support environment, also called MOTION. This environment consisted of three layers: communication layer, teamwork support layer, and user-interface layer. The goal of the environment was to enable application developers to build domain-specific or task-specific team-support software tools (or groupware). For example, one pilot application was built to support product design review meetings. Such

meetings have similar requirements to the ones we have described for program committee meetings.

The communication layer is based on a peer-to-peer architecture and provides event-based communication; the user-interface layer provides facilities for building GUI interfaces that are device-independent; the teamwork support layer (TWS) exposes the main facilities of the platform to application builder through an application programming interface (TWSAPI). The primary abstractions provided by the TWS are users, artifacts, profiles, communities, repositories, and access rights. Users define specific actors in the process; artifacts define objects to be accessed and manipulated by users; communities define groups of users with similar interests; profiles define attributes (meta-data) of users, artifacts, and communities; repositories define (distributed) containers for artifacts; access rights define which actors may access which objects. Because the underlying architecture is peer-to-peer, repositories may be formed from sub-repositories located on different peers. Because MOTION supports off-line processing, repositories may sometimes not contain all data if some peers are unavailable. With the use of these abstractions, we can begin to model the Program Committee Review problem as shown below:

- Users: Authors, Reviewers, Chairs, External Reviewers
- Community: Reviewers, Chairs, Reviewers for specific papers (subcommunity of Reviewers)
- Artifacts: Papers, Reviews, Notification letters to authors (automatically generated), Proceedings
- Profiles: Describe reviewers, their expertise and interests; papers and reviews can also have profiles
- Access rights defined by chairs as a result of paper assignment and conflict declarations
- Repositories: for Submitted papers, for Reviews, for Camera-ready papers

Communities share artifacts. For example, a (sub)community of reviewers assigned to a paper may be associated with that paper. They may subscribe to be notified of any changes in the status of the paper, such as submission of new reviews for that paper. The publish-subscribe communication paradigm of MOTION integrates nicely with the community notification process. The peer-to-peer architecture of MOTION provides natural support for ad-hoc networking so that a program committee meeting can be run on an ad-hoc network of those members present at the meeting. The collection of artifacts available on the laptops of the members in the network forms the global repository of the review process. MOTION provides no support for process definitions. That is, the support for tasks such as assigning papers to reviewers must be programmed in a programming language using the API of TWS to make use of TWS abstractions. In practice, because of the high-level nature of the TWS abstractions, these programs turn out to be quite short and easy to write. Still, the knowledge about the process is embedded in these programs and relationships among these tasks are not evident at all.

We have concluded that MOTION needs a process definition and composition component to enable tasks to be defined and executed (enacted). Such a process definition component would make it possible to document, instantiate, and reuse task definitions. For example, a generic task could be "assigning tasks". Two specific instances of this task are "assigning a review to a reviewer" and "assigning resolution of divergent reviews to a community of reviewers." Given appropriate task definition units, a given program committee review system could be built by instantiating and composing appropriate units into workflows. A workflow engine can then enact and monitor the progress of the process. Some process composition ideas are being explored in the DMC architecture presented in [Dustdar & Gall].

We can start the process of task definition by identifying the many tasks embedded in the process. Some tasks may be defined as individual and others as group tasks. For example, reviewing a paper is an individual task; discussing the reviews is a group task. Some tasks are specific, such as reviewing a paper; other tasks are more general but also not as well defined such as searching for specific expertise and skills among the reviewers. Over time, we could build a repository of such task units that can be used in future projects.

A particularly useful task unit is "search for an expert." We first encountered the need for this activity in the MOTION project where both of our industrial partners presented it as a critical problem for them. When faced with a critical problem, it is often difficult to find who holds the critical knowledge for solving that problem and where that person is. Apparently, large corporations have this problem and it is increasingly in importance as work becomes knowledge intensive and knowledge workers become mobile. We have seen several instances in the program committee problem where such a task would be needed: in finding members for the program committee initially and when trying to assign reviewers to papers. We will see other examples in the next section.

## 5. What about pervasive services?

What I have described as the process of paper reviewing is typical of review processes. It is also typical of ad-hoc, mobile teams of knowledge workers. Indeed, for most of the members in the committee, the task of reviewing papers is only a small part of their daily duties. Typically, they perform the task in their "spare" time and often while traveling. They are collaborating in this task with other members who are part of other organizations and also involved in this task in their spare time. The members do not share any common computing infrastructures. In many cases, they may not even have met in person. The almost exclusive mode of communication is email. Most members do their reviewing off-line and submit their reviews when they can establish a connection to the Internet. They would probably prefer to submit their reviews off-line and upload them (automatically) when connected. Most systems do not support such a feature today. Therefore, off-line review preparation and on-line cut-and-paste is common practice.

Let us first look at some obvious pervasive computing facilities that can directly influence our program committee meeting process. The first service that one expects is

"presence awareness." If we could detect which members were present in the meeting, we could improve many of the processes involved in the meeting. For example, the system could notify those members with conflicts of interest to exit the room just before discussion of the paper starts. It could also notify the committee when discussions may begin. The system could maintain a list of papers that may be discussed to ensure that no paper discussion starts unless all the relevant reviewers are present. It could even check the presence of other reviewers who are expert in the area, even if they are not reviewers of the paper. It could identify reviewers who could shed light when disagreements arise. This idea can be generalized to a meeting agenda manager that dynamically updates the agenda on the basis of presence of individuals. A more sophisticated version would also use a policy module for ordering the discussions. Presence detection could also be used to notify a particular group who may have a review to discuss among themselves when they are in the vicinity of each other. (Remember that sometimes the people may not know each other personally.)

Another obviously useful service is support for information management. Information management includes enabling access to information, filtering relevant information, and synchronizing the information in the face of updates. The environment should be able to provide such features on the basis of the location of the person and the device he or she is carrying, without the need for painstaking device driver or plugin installations. Data synchronization is an especially important issue. This is necessary both after users have been off-line and after changes have been made by other people. For the reviewers in our scenario, this means that the status of their papers are kept up to date on their local device. A simple synchronization would be to update the clock on all devices when entering a different time zone. The detection of the presence of a person at a capable site can trigger relevant synchronization processes automatically.

Presence detection can also be the basis for establishing communications of relevance. We have seen that at least two kinds of communication are necessary: notifications and messages of longer length. In fact, notification services seem indispensable to mobile workers even though they are not so universally supported. Pervasive computing services should make this easier in the future.

We can divide further help that can come from additional (or better) technology—such as pervasive computing and services—into generic support for knowledge work activities and specific support for program committee meetings. The generic support needed is to help knowledge workers focus on their current activity, help them in carrying out that activity, and ease the task of switching between activities. To focus on the current activity, the environment must present the context appropriate for that activity. For example, if I am involved in reviewing papers for a particular conference, my work context should contain the papers assigned to me and the status of each paper, but also any notes I have made about those papers, any emails I have received about the reviewing activity, perhaps the conference's Website, past proceedings of the conference, copies of the authors' previous relevant papers, and the paper's cited references. At the moment of working on this activity, I would not like to see any of my other email messages or receive any other email unless it is related to this task (or is an urgent message). I would like to receive a notification if some

information about a paper I am reviewing has changed, for example, if the PC Chair has decided that the paper should be discarded due to some discovered irregularity.[1]
Supporting the current activity requires an encoding of the task being performed so that the environment can monitor and report the progress of the activity. This also requires maintaining the context of the activity, such as all needed and affected documents, relevant team members, those depending on the activity and those contributing to the activity, messages communicated in relation to this activity, schedules, and relationship to other activities. For example, in reviewing a paper, a PC member is pursing an activity that is a part of a larger activity including a group of reviewers working on the same paper, whose group activity is itself part of a larger activity initiated by the PC Chair, and so on.
Switching between activities requires the suspension and saving of the current context and reloading and restarting of a new context, preferably with a convenient overview of the current status that includes all changes and messages since the activity was suspended.
All this support for activities must be provided taking into account the following realities of the knowledge workers' environment:

- Complete distribution: people are distributed around the world with access to their private computing environment without a centralized shared facility
- Complete heterogeneity: people use a variety of hardware devices and software tools; there is increasing variety of special-purpose and general-purpose devices
- Mobility and disconnected operations: people are often away from their home location (if indeed they have one) and sometimes are not able to maintain an Internet connection

On the other hand, what we will have are environments with rich computing and communication capabilities and knowledge workers traveling between them carrying their various devices. Since each device cannot carry the entire database or application, support for the application must be embedded in these environments. We can classify the support needed for the tasks we have discussed as information management, communication management, and workflow management. All of these must be managed on a global scale. Traditional centralized or client-server architectures are clearly insufficient to meet these requirements. Entirely new software architectures and concepts will be needed. We envision that applications will be composed of units that are spread over pervasive computing environments and devices that are carried by people. Devices carry the various task contexts and switch between them with the help of the services provided by the environment. In this way, users do not carry specific documents or data with them. These will follow them based on their task context

[1] Even though we seem to be unable to deal with unwanted spam mail, the filtering of relevant messages appears to be an easier problem to solve.

as needed. The paradigm is that as people move from one rich environment to another, they recharge their contexts just as they may recharge their batteries.

## 6. Conclusions

I have described a scenario of a teamwork process in some detail. I believe that this scenario can be used for identifying teamwork processes and direct the search for supporting services. I have also considered how such problems may be addressed in future pervasive computing environments. I have only scratched the surface of this rich field. In my own group, we are working on architectural design issues [Dustdar&Gall], heterogeneity issues [Gschwind, Oberleitner, Jazayeri], and correctness of event-based applications [Fenkam, Gall, and Jazayeri]. I believe that correctness and quality are fundamentally important issues in an increasingly wired world and any software support, including those for teamwork support, must come with proven guarantees about their offered services. With pervasive services, guarantees for the support of heterogeneity and maintenance of security and privacy guarantees are required qualities.
The process and teamwork issues are studied in many communities, including workflow, CSCW, and software process. References may be found from the MOTION Website. There are many projects on pervasive computing environments, including Aura at Carnegie Mellon University and Gaia at the University of Illinois. We are examining those very carefully. Our own ideas on composing applications out of mobile components are presented in [Jazayeri].

## 7. References

Schahram Dustdar and Harald Gall, “Architectural Concerns in Distributed and Mobile Collaborative Systems,” Journal of Systems Architecture, to appear.

Pascal Fenkam, Harald Gall, and Mehdi Jazayeri, “Composing Specifications of Event Based Applications,” Proceedings of the Conference on Fundamental Approaches to Software Engineering (FASE 2003), Springer Verlag, April 7-11, 2003, Warsaw, Poland.

Thomas Gschwind, Johan Oberleitner, and Mehdi Jazayeri, “The Vienna Component Framework: Enabling Composition Across Component Models,” Proceedings of the 25th International Conference on Software Engineering, pp. 25-35, May 3--10, 2003, Portland, Oregon, USA.

Mehdi Jazayeri, “Software Components: Pervasive Challenges (invited paper),” Proceedings of Radical innovations in Software Engineering of the Future (Monterey 2002), Venice, Italy, October 2002.

Oscar Nierstrasz, “Identify the Champion,” Pattern Languages of Program Design, N. Harrison, B. Foote and H. Rohnert (Eds.), vol. 4, Addison Wesley, 2000, pp. 539-556.

Harald Gall, Engin Kirda, Pascal Fenkam, and Gerald Reif, "TWSAPI: A Generic Teamwork Services Application Programming Interface," Proceedings of the International Workshop on Mobile Teamwork 2002, Collocated with the 22nd International Conference on Distributed Computing Systems(ICDCS 2002), July 2-3, 2002, Vienna, Austria.