Peer-to-Peer Remote Conferencing

Fabio Calefato, Filippo Lanubile, Teresa Mallardo
Dipartimento di Informatica,
University of Bari
{*calefato | lanubile | mallardo*}*@di.uniba.it*

Abstract

Global software development (GSD) is nowadays pervasive among large enterprise organizations. Physical separation in GSD has raised many issues, mainly due to cross-sites communication and coordination problems, which have made software development an even more challenging task. Hence, distributed workgroups need tools to support a load of activities that usually take place through the direct interaction among people. This paper presents a tool, called P2PConference, to conduct conferences over a distance. The tool provides basic features for simple brainstorming sessions as well as more sophisticated features to accommodate the needs of other types of meetings, such as presentations and panels. P2PConference adopts a decentralized architecture and it is implemented upon a peer-to-peer infrastructure platform, called JXTA.

1. Introduction

Over the last few years, large enterprise organizations have embraced global software development distributed over multiple geographical sites [9]. Communication is the core function of cooperation that allows information to be exchanged between team members. Distance has a negative effect for communication-intensive tasks, such as software design, and on spontaneous conversation [8], where people informally communicate valuable pieces of information.

Distance is usually offset by Internet-based technologies: globally distributed workgroups typically rely on centralized systems, mostly built on top of web-based development platforms, to support collaboration across time and space. However, peer-to-peer (P2P) applications, based on a decentralized architecture, are increasingly becoming popular to exchange instant messages, share common information and applications, and jointly review/edit documents. Collaborative P2P applications exhibit the following advantages with respect to client-server counterparts:

- Autonomy. In a P2P system every peer is an equal participant while being a final authority over its local resources. In this way everyone can share information but, at the same time, can pose restrictions on confidential data through access rights management and data encryption. When enterprise data are distributed on many places and on different devices, P2P systems can provide an easier and cheaper alternative to enforcing a convergence into a centrally managed data repository.
- Intermittency. P2P systems are designed by giving for grant that any peer can disappear at any time because of network disconnections, either deliberate or accidental. P2P collaborative systems use resource replication and different synchronization mechanisms, based on proxies for sending/receiving messages in the network on behalf of the disconnected sender/receiver. In this way, users can work to shared content even when offline and automatically propagate changes at the first reconnection.
- Immediacy. P2P applications have shown themselves able to support direct exchanges between peers, as in the case of instant messaging. P2P collaboration systems, based on near real-time communication mechanisms and synchronous presence of the peers, can provide immediate responses by participants to enable effective person-to-person interaction.
- Cost lowering and compartment. P2P systems are valuable means to lower infrastructure cost by using existing infrastructure and distributing the maintenance costs. Centralized systems that serve many clients typically bear the majority of the cost of the system. When the cost becomes too large, a P2P architecture can help spread it over all the peers.

Under these conditions, a P2P collaborative infrastructure can complement or even replace client-server platforms for the creation of ad-hoc or small workgroups, drastically reducing the cost of infrastructure setup and ownership. Due to P2P own features, it is possible to quickly establish dynamic collaborative

groups, composed of people from different organizations accessing shared resources and interacting in a near real-time manner.

This paper presents P2PConference, a P2P remote conferencing tool which has been developed at the University of Bari. In the next sections, we first introduce the underlying platform and then describe how the tool works. In the last section, we show how the tool is evolving.

2. JXTA

P2PConference has been developed using the Java implementation of JXTA [10], a network programming and computing platform for P2P systems. Project JXTA was originally conceived by Sun Microsystems and designed with the participation of a small number of experts from academic institutions and industry. The platform was released as an open source project early in the 2001 to become the standard foundation for P2P systems.

The project had to address some issues that were set as objectives [7]:

- Interoperability. Nowadays there are several P2P systems that, though offering the same services (e.g. file sharing), are incompatible because of the lack of a common infrastructure. This issue is referred to as *danger of fragmentation* [14]. JXTA aims at becoming the missing standard and, hence, it has been proposed to IETF [12].
- Platform independence. No target platform (as both programming language and operative system) has been chosen to develop JXTA, thus to embrace a larger base of developers and final users.
- Ubiquity. JXTA has been designed to be implemented on a wide range of digital devices, from cell phones to servers.

At the highest abstraction level, JXTA is a set of six protocols, each defined by XML-based message exchange:

- Peer Discovery Protocol (PDP)
- Peer Revolver Protocol (PRP)
- Peer Information Protocol (PIP)
- Peer Membership Protocol (PMP)
- Pipe Binding Protocol (PBP)
- Endpoint Routing Protocol (ERP)

JXTA technology is designed to provide a layer on top of which other services and applications are built (see Figure 1). Typical P2P software stacks break down into three layers. The lowest level (referred to as JXTA core) deals with peer establishment, communication management, such as routing. In the middle (JXTA services) the layer provides higher level services, such as indexing, searching and file sharing, built upon the low-level features of the core. At the top is the layer of applications (JXTA applications): any P2P system built using the services beneath.

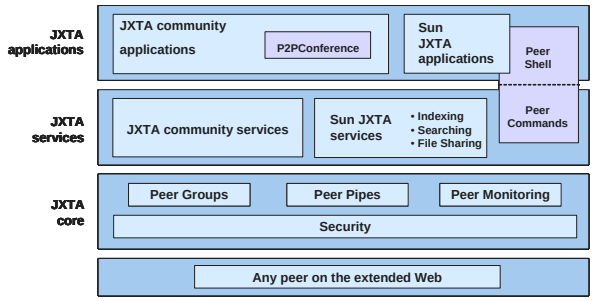


Figure 1. The layered architecture of JXTA

3. P2PConference

P2PConference was inspired by the eWorkshop tool [1] from CeBASE [5]. eWorkshop is a simple web-based collaboration tool to organize and conduct remote, text-based meetings with the aim of gathering and synthesizing knowledge from a group of invited experts. However, P2Pconference is not a mere porting of eWorkshop onto the JXTA platform. Other than replicating the basic features of eWorkshop, we have added new capabilities to run different types of remote conferences, and allow organizers to exercise more control on the participants.

The primary functionality provided by the P2PConference is a closed group chat with agenda, whiteboarding and typing awareness capabilities. The tool allows participants to communicate by typing statements that will appear on all participants' message boards. By responding to statements on the message board, they can carry on a discussion on-line. Around this basic feature, we built other features to help organizers control discussion.

The organization of a remote conference (or simply conference, hereafter) follows a strict protocol which mandates the organizers to choose the main discussion topic, schedule the meeting and decide whether or not to run training sessions (to let participants try out the tool), and, finally, send invitations to participants by e-mail.

Most participants in a conference are experts in their respective domain. Organizing a new conference implies to set up a support team, which consists of the following roles: moderator, director and scribe.

The *director* is the actual conference organizer, since he/she is supposed to choose the main discussion topic and the items that it is composed of, schedule the conference and send invitation e-mails, which contain an user id and password to join the discussion.

The *moderator* is responsible for monitoring and focusing the discussion (e.g. proposing items on which to vote) and maintaining the agenda. Among the support team members, only the moderator is an active participant in the sense that he contributes actual responses during the meeting. He/she is also responsible for assessing and setting the pace of the discussion, that is, he/she decides when it is time to redirect the discussion onto another item.

As the discussion moves from one item to another, the *scribe* captures and organizes the results displayed on the whiteboard area of the screen. When the participants have reached a consensus on a particular item through a vote, the scribe summarizes and updates the whiteboard to reflect the outcome. The content of the whiteboard becomes the first draft of the meeting minutes.

The tool screen has five main areas: agenda, input panel, message board, whiteboard, and presence panel (see Figure 2).

The *agenda* is managed by the moderator and indicates the status of the meeting ("started", "stopped") as well as the current item under discussion.

The *input panel* enables participants to type and send statements during the discussion.

The *message board* is the area where the meeting discussion takes place. Statements are displayed sequentially, tagged with the time of when they were sent and the sender's name.

The *whiteboard* is used to synthesize a summary of the discussion and is controlled by the scribe. In order to realize the goal of measuring the level of consensus among the participants, all of the items added to the whiteboard are subject to voting announced by the moderator. When participants do not agree with how the statements on the whiteboard were formulated, negotiations initiate in order to come up with a more accurate description of the results of the discussion.

The *presence panel* shows participants currently logged in and the played role.

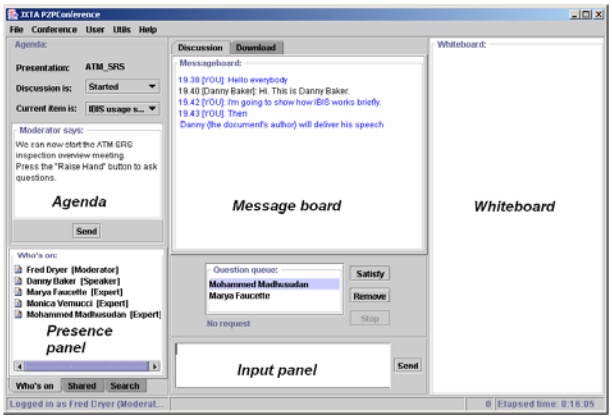


Figure 2. P2PConference screenshot

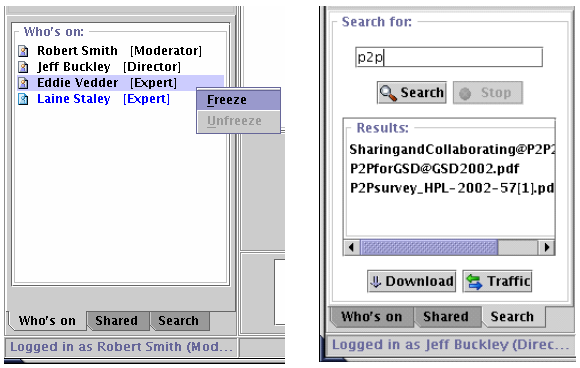
All of these features can also be found in the eWorkshop tool. We further enhanced support for remote conferencing by adding the following features:

- Control. Conference organizers need more control power over participants. Hence, we also added *freezing* − moderator can freeze those experts who disturb, forbidding them to type and ensuring the discussion to flow smoothly (see Figure 3a) − and *hand raising*, that is participants must ask the moderator the right to talk or ask questions.
- File sharing. A collaborative tool cannot be such without file sharing capability (see Figure 3b).
- Protection. A conference is said to be "protected" if it does not allow users to access the drafts (i.e. the discussion log and the whiteboard content) saved by the peer into HTML files. The only participant allowed is the director. This option ensures the organizers that no one else can carry on a conference analysis.

Indeed, the presence of the moderator only prevents the discussion to become unconstrained, ensuring that all of the items in the conference agenda are discussed. This kind of remote meeting is apt for brainstorming sessions with limited or no control over the participants for the organizers. We did not want to bind the organizers to run only brainstorms and, hence, we identified three different types of existing conferences to model and implement in P2PConference:

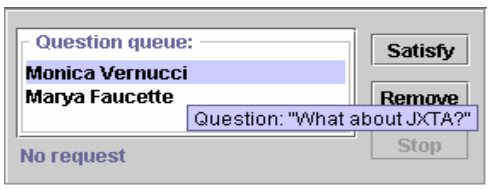
- Meeting. It ensures a limited control power since the moderator can only "freeze" disturbing participants (i.e., the moderator may forbid them to type and send statements). This conference type models simple, remote brainstorms.

- Presentation. This is a more complex kind of conference: one special invited expert, the *speaker*, delivers his own speech and the other invited experts (i.e., the audience) can ask him/her questions, after "raising their hands". The moderator manages the queue of the asked questions (see Figure 4a).
- Panel. It is a generalization of presentation: there is more than one speaker, the so-called *panelists,* and, since any of them can deliver a speech, they have to request the right to speak by "raising their hands" (see Figure 4b). Moreover, the experts who want to ask a question are to pick the panelist(s) and raise their hands too. Hence, the moderator manages two separate queues, one for the panelists and one for the experts

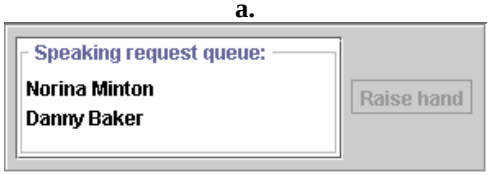


a. **b.**

Figure 3. The presence panel with freezing menu (a) and the search panel (b)



a.



b.

Figure 4. Hand raising panels for question requests (a) and speaking requests (b)

5. Conclusions and Further Work

In the field of collaborative software development (CSD) environments P2P technology and decentralization have begun to being introduced [2, 3].

In this paper we have described P2PConference, a tool for running remote conferences. The tool is also an open-source software hosted at the Project JXTA site [11]. Currently, one of the authors has the role of project owner, two fifth-year computer science students act as developers (committers), and thirteen people are contributors (mainly for issue reporting and bug fixing).

Much of the tool functionality has been implemented in the first release. Also, we plugged P2Pconference into IBIS [13], a tool developed at the University of Bari to support software inspections for geographically dispersed teams. Using Java Web Start [15], inspectors can launch P2PConference and run a kickoff meeting to provide background information on the inspection process or the product being inspected.

Current work is aimed to make deployment easier, by automating the initial peer configuration, and add support for presentation sharing and co-browsing. As further work, we are planning to develop a remote-conferencing plugin to integrate our tool in an extensible IDE, such as the Eclipse Platform [6].

6. References

[1] V. Basili *et al.*, "Building an Experience Base for Software Engineering: A report on the first CeBASE eWorkshop", *Proc. of International Conference on Product Focused Software Process Improvement* (PROFES 2001), Kaiserslautern, Germany, September 2001, pp 110-125.

[2] bitkeeper.com, *BitKeeper Source Management*, http://www.bitkeeper.com/Products.BK_Pro.html

[3] S. Bowen, and F. Maurer, "Using peer-to-peer technology to support global software development – some initial thoughts", *Proc. of the Int. Workshop on Global Software Development (ICSE 2002)*, Orlando, FL, USA, May 2002.

[4] G. Canfora, F. Lanubile, and T. Mallardo, "Can Collaborative Software Development Benefit from Synchronous Groupware Functions?", *Proc. of the 2nd Workshop on Cooperative Supports for Distributed Software Engineering Processes* (CSSE 2003), Benevento, Italy, March 2003.

[5] CeBASE Web Site, http://www.cebase.org

[6] eclipse.org, Eclipse Foundation website, http://www.eclipse.org

[7] L. Gong, "JXTA: A network programming environment". *IEEE Internet Computing*, 5(3):88--95, May-June 2001.
[8] J. D. Herbsleb and R. E. Grinter, "Architecture, Coordination, and Distance: Conway's Law and Beyond", *IEEE Software*, Vol. 16, No. 5, September/October 2001, pp. 16-20, pp.63-70.
[9] J. D. Herbsleb and D. Moitra, Global Software Development, IEEE Software, Vol. 18, No. 2, March/April 2001, pp. 16-20. Software Engineering, *IEEE Software*, Vol. 19, No. 3, May/June 2002, pp. 26-38.
[10] jxta.org, *Project JXTA Home Page*, http://www.jxta.org
[11] jxta.org, *P2PConference Home Page*, http://p2pconference.jxta.org
[12] jxta.org, *IETF standardization effort*, http://www.jxta.org/IETFStandard.html
[13] F. Lanubile, and T. Mallardo, "Tool Support for Distributed Inspection", *Proc. of International Computer Software and Applications Conference (COMPSAC 2002)*, Oxford, UK, 2002.
[14] D.S. Milojicic *et al*. "Peer-to-Peer Computing", HP Laboratories Palo Alto, March 2002
[15] sun.com, *Java Web Start Technology*, http://java.sun.com/products/javawebstart