A DECENTRALIZED CONFERENCING TOOL FOR AD HOC DISTRIBUTED WORKGROUPS

Fabio Calefato, Filippo Lanubile [1])

Abstract

Open source software (OSS) projects are an extreme case of geographically distributed development, with individual developers scattered throughout the world. In addition to relying upon asynchronous communication tools, a number of OSS project members use chat-based communication tools, mainly for bug triage and rapid decision-making about project evolution. However, simple chats do not fit well the needs of structured meetings.

This paper presents P2PConference, a tool to conduct text-based 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 requires no infrastructure and administration costs since it adopts a decentralized architecture and it is implemented upon the JXTA peer-to-peer platform. The decentralized architectural solution makes the tool well suited to support ad hoc distributed workgroups.

1. Introduction

Distributed software development has become mainstream among large enterprise organizations, with software projects split over multiple geographical sites [17]. Open source software (OSS) projects are an extreme case of geographically distributed development, as team dispersion is the highest possible. In OSS projects individual voluntary developers work in arbitrary locations, rarely meet face to face, if at all, and coordinate their activity typically by means of email and bulletin boards [29].

OSS projects extensively use mailing lists for a wide range of purposes (e.g., announcements and user support), also because emails can be automatically archived and hence provide a trail of decision-making and an historical rationale for the project [10]. However, real time communication is better suited for supporting rapid decision making because of no inherent latencies. While chat-based

[1] Dipartimento di Informatica, University of Bari, email: [calefato|lanubile]@di.uniba.it

discussions can be considered a surrogate for informal communication or “watercooler” conversations, they do not resemble the structured-meeting communication style, as IRC lacks features to support moderated discussions with different participant roles.

Distributed workgroups typically rely on centralized systems, mostly built on top of web-based development platforms, to support collaboration across time and space. Earliest Collaborative Software Development (CSD) platforms were developed within OSS projects [3]. 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.

P2P is a self-organizing networking paradigm: peers are self-administered entities that autonomously discover each other, arranging themselves into a logical overlay network, which does not rely upon centralized services such as DNS and routing. Collaborative P2P applications exhibit the following advantages with respect to client-server counterparts: autonomy, intermittency, immediacy, and cost lowering and compartment.

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 leveraging existing resources 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 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 developed at the University of Bari. Our goal is to use the tool to run structured, text-based conferences over a distance in distributed collaborative scenarios, with no administration and infrastructure costs since it is implemented upon a decentralized platform. In the next sections, we first introduce the underlying platform, then describe how the tool works and, we discuss opportunities of using decentralization to support synchronous communication in ad hoc collaborating environments. In the last section, we draw conclusions and show how the tool is evolving.

2. JXTA

P2PConference has been developed using the Java implementation of JXTA [22], 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 [13]:

- 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* [28]. JXTA aims at becoming the missing standard and, hence, it has been proposed to IETF [21].
- 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.

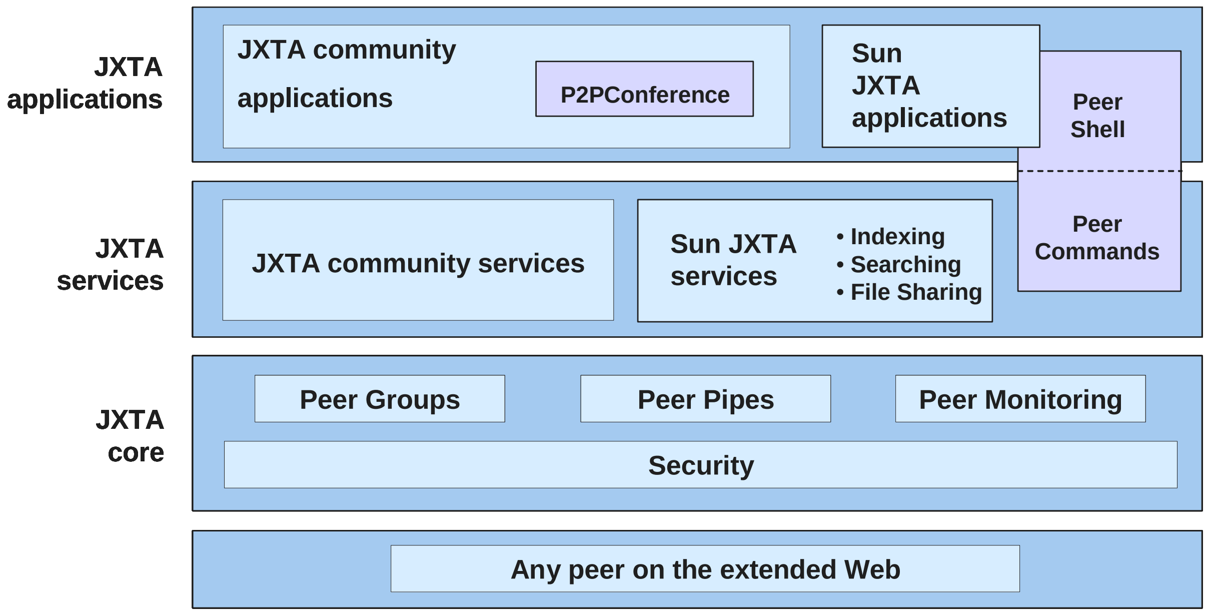


Figure 1. The layered architecture of JXTA

3. P2PConference

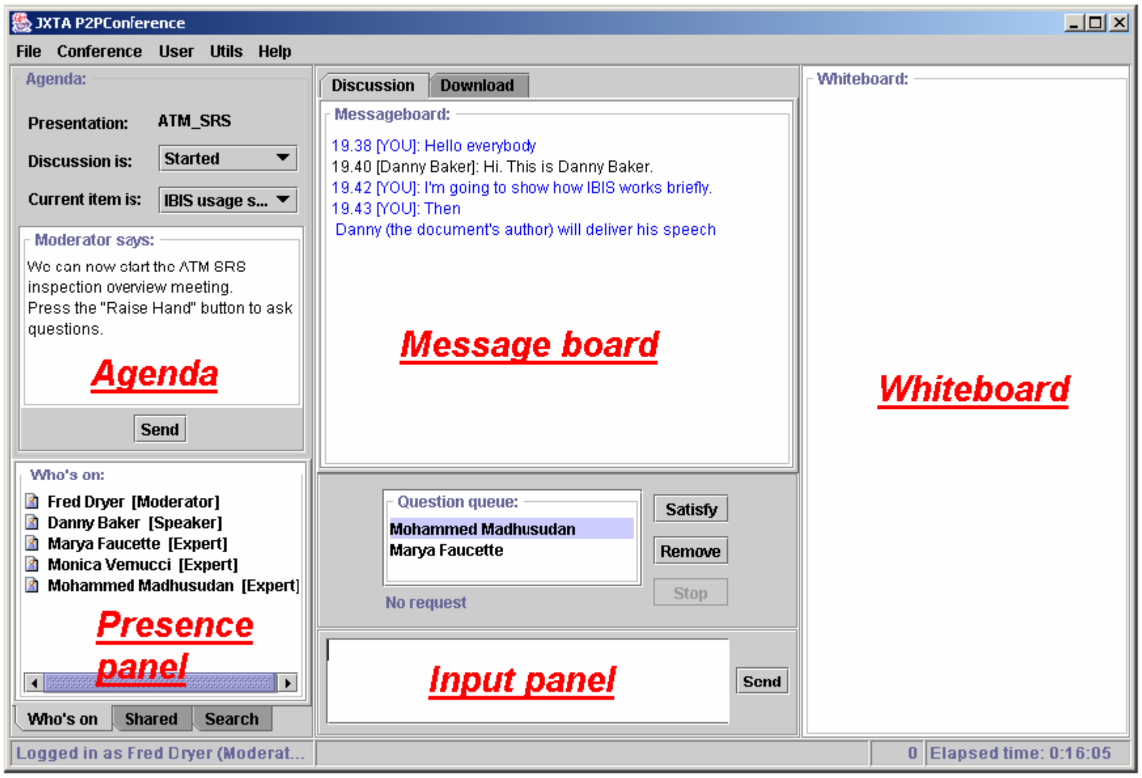
P2PConference was inspired by the eWorkshop tool [1], a web-based collaboration tool which is used in CeBASE [4] to organize and conduct remote, text-based meetings with the aim of gathering and synthesizing knowledge from groups of invited experts. However, P2Pconference is not a mere porting of eWorkshop 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 an online discussion. Around this basic feature, we have 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/she may contribute actual responses during the meeting (this is an option that can be selected during conference planning). 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. The scribe is not supposed to take part actively in the conference, so he/she can keep focused on the discussion flow [14]. When 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.



Presentation: ATM_SRS
Discussion is: Started
Current item is: IBIS usage s...

Moderator says:
We can now start the ATM SRS inspection overview meeting.
Press the "Raise Hand" button to ask questions.

19.38 [YOU]: Hello everybody
19.40 [Danny Baker]: Hi. This is Danny Baker.
19.42 [YOU]: I'm going to show how IBIS works briefly.
19.43 [YOU]: Then
Danny (the document's author) will deliver his speech

Fred Dryer [Moderator]
Danny Baker [Speaker]
Marya Faucette [Expert]
Monica Vemucci [Expert]
Mohammed Madhusudan [Expert]

Mohammed Madhusudan
Marya Faucette
No request

Logged in as Fred Dryer (Moderat...
0 Elapsed time: 0:16:05

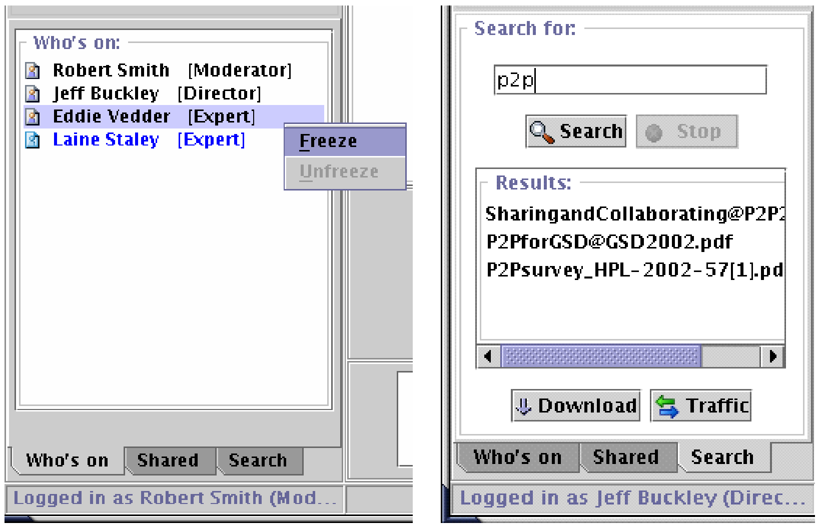
Figure 2. P2PConference screenshot

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*). This feature has been implemented using the CMS community project 0[24]. Users that join a conference can share whatever file they own and search for files shared by others in the same conference group. However, since a conference resembles a short-term shared workspace, the share-and-search approach might not fit well. Based on the approach followed by the Jxtacast project 0[25], we might rather build a file repository, so as to have shared files propagated to either the entire group or selected conference participants.

- 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.



a. **b.**

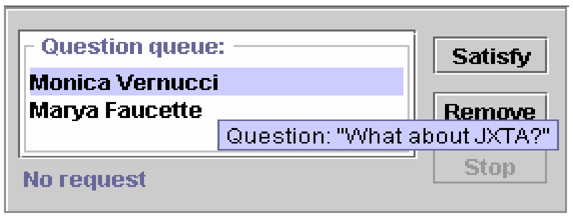
Figure 3. The presence panel with freezing menu (a) and the search panel (b)

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:

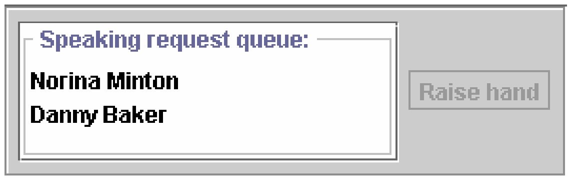
- Meeting. It ensures a limited control power since the moderator can only “freeze” disturbing participants (i.e., the moderator may forbid them to type and send statements). This conference type models simple, remote brainstorms.

- Presentation. This is a more complex kind of conference: one special invited expert, the *speaker*, delivers his own virtual, text-based speech and the other invited experts (i.e., the audience) can ask 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 4. Hand raising panels for question requests (a) and speaking requests (b)

4. Discussion

OSS projects are mainly the effort of voluntary contributors from different countries. However, several companies have been subsidizing the development of OSS projects, like GNOME or Linux, with the hope of accelerating their growth by donating infrastructure support and employing full time workers [9]. If the sponsor company removes its funding, the project is to cope with this loss and the risk of being abandoned. Hence, OSS projects should promote the use of process and tools that grant continuity [8].

Distributed collaborative environments, and OSS projects in particular, can benefit from the exploitation of decentralized P2P architectures. Using decentralized tools, such as distributed source code management (SCM) and bug repositories, would help manage this risk. Torvald’s choice of BitKeeper (BK) [2] in maintaining the Linux-development tree [27] is consistent with this scenario.

BK, unlike CVS, facilitates distributed development as it grants developers write access to their own repositories, not the main central one. Hence, BK creates a network of maintainers in which individual developers need to rely on and trust each other in order to submit their patches to or merge with other developers' trees.

Many OSS projects require that large changes be discussed before implementation starts, thus allowing developers to gather feedback on proposed strategies. This usually happens by posting emails to a specific discussion mailing list. Yet, asynchronous discussions introduce a delay factor [16]. Conventional face-to-face practices are essential for successful project collaboration, as they contribute to build trust among members [19] and increase team cohesion [30]. One approach to overcome distance barriers is to offset them by getting together during those activities that require intensive interaction and coordination, such as project startup or requirements negotiation, or by temporarily collocating teammates or periodically travelling for cross-site visits. However, this is often unfeasible in globally dispersed projects and it seems impossible in OSS project scenarios. Instead, new information and communication technology solutions are needed to support distributed collaboration.

In addition to relying exclusively upon asynchronous communication, some OSS projects use a variety of synchronous communication tools. Real time communication is better suited for supporting rapid decision-making because of no inherent latencies, unlike threaded email discussions. GNOME Board of Directors meets every week via phone conferencing to discuss various topics pertaining to the GNOME Foundation and the GNOME project [11]. In [6] the author describes the issues he has personally encountered in distributed projects, working with impersonal media and email in particular. Long and heated email discussions result stressful because, due to the serial nature of the medium, participants cannot interrupt each other until the whole message is dispatched. The other consequences of exclusively using email for discussion are the difficulty of coming to conclusions or reaching consensus, the lack of meeting minutes to review after being out of touch for any length of time and the absence of a moderator that monitors the discussion and tries to keep it flowing in productive ways. Moreover, when using email in the decision-making process problems arise because people tend to "expect everyone to be as responsive – or at least to overlook the possibility that they aren't – and to assign deadlines and decision points accordingly."

Success factors of big OSS projects have been identified in [29]. The analysis showed that OSS projects would fail without a large, critical mass of users who contribute reporting bugs and submitting patches. GNOME sub-projects leaders, bug reporters and other contributors join the IRC #bug channel hosted on irc.gnome.org and discuss on how to close fixed bugs, confirm the newly entered ones, making sure that enough correct information has been provided, or remove them. Bug Days are a valuable means to have a synchronous, near real-time discussion among software module leaders and bug reporters without the inherent latencies of threaded email messages.
Synchronous communication is effective only when participants are located in similar time zones. If not all participants can take part in synchronous communications, it is essential that some archival of the communications be saved, otherwise key participants may be left out of a critical discussion. This is why the minutes of the GNOME synchronous meetings are posted publicly on the foundation-list archives. Nevertheless, Bug Days have proved so effective that JXTA community members have borrowed the idea to run *JXTA Bug Days* with the same purpose [20]. Provided that channels are

hosted on free IRC servers, IRC chat is used because there are neither infrastructure nor administrative costs, as participants simply run an IRC client and join a given channel.

Simple chat-based communication is not as effective as the structured communication achievable using more sophisticated conferencing tools. There is a variety of conferencing tools, mostly web based, such as Centra [5] and WebEx [32], whose features include agenda capability, moderated discussion with different participant roles, speaking in turn by hand raising, presentation sharing and co-browsing. These tools support a more structured discussion which resembles facilitated workshops [14] rather than unstructured meetings. However, web conferencing tools, apart from being not free, may require infrastructure or hardware upgrades to fully support audio/video communication. Moreover, audio conferencing may not always fit some participants because of language barriers. Communication issues become even more severe in audio conferencing. Whilst English is the de facto standard for collaborating across borders, participants who do not speak the language fluently may stay quiet during audio conferences, neither raising issue, nor helping to resolve them. Instead, non-native English speakers are typically better at reading and writing. Hence, structured text-based discussions, directed by a formal leader, can give anyone the chance to fully participate [15]. Audio conferencing is a viable option only when all participants do share a common language with small disparities.

Analogously to web conferencing tools, P2PConference provides features like whiteboarding, agenda and hand raising, to accommodate the needs of structured meetings. However, P2PConference does not require any hardware upgrade since it is implemented upon a P2P overlay network, nor it adds any network administration costs, as P2P is a self-organizing networking paradigm by definition. Currently, the tool does not support audio conferencing yet. However, we look forward to integrate it using libraries developed by other JXTA community projects.

5. Conclusions and Further Work

P2Pconference can be used to support collaboration in distributed software development environments, especially to support ad-hoc and dynamic workgroups when infrastructure and administration costs are hardly bearable for the adoption and usage of a conferencing tool only in the short term. It could also be used in smaller, non-funded OSS projects: one could think to run bug triage sessions using P2PConference and have project participants joining the virtual meeting with their peers, bugs scheduled as items in the agenda list, file sharing, a moderated discussion and a trial of the meeting in the whiteboard.

P2Pconference has been deployed as an open-source software hosted at the Project JXTA site [23]. Currently, the first author 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. P2Pconference has already been plugged into IBIS [26], a tool developed at the University of Bari to support software inspections for geographically dispersed teams. Using Java Web Start [29], 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 integrate our tool as a plug-in into an extensible IDE, such as the Eclipse Platform [7], and add audio support. Finally, we intend to conduct repeated case studies, in the context of student project works, to support communication-intensive activities such as requirements elicitation and design reviews. This will allow us to collect experimental data useful for an early evaluation of tool effectiveness.

Acknowledgments

We would like to thank all P2PConference project participants for their contributions.

References

[1] BASILI, V. et al., Building an Experience Base for Software Engineering: A report on the first CeBASE eWorkshop, Proc. of Int. 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] CANFORA, G., LANUBILE, F., MALLARDO, T., Can Collaborative Software Development Benefit from Synchronous Groupware Functions?, Proc. of the Workshop on Cooperative Supports for Distributed Software Engineering Processes (CSSE 2003), Benevento, Italy, March 2003.

[4] CEBASE.ORG, CeBASE web Site, http://www.cebase.org

[5] CENTRA.COM, Centra web site, http://www.centra.com

[6] COAR, K., The Sun Never Sets on Distributed Development, ACM Queue 1(9):32-39, December/January 2003-2004

[7] ECLIPSE.ORG, Eclipse Foundation web site, http://www.eclipse.org

[8] ERENKRANTS, J.R., TAYLOR, R.N., Supporting Distributed and Decentralized Projects: Drawing Lessons from the Open Source Community, Proc. of Workshop on Open Source in an Industrial Context (OSIC 2003), Anaheim, California, USA, October, 2003.

[9] GERMAN, D.M., The evolution of the GNOME Project, Proc. of the Workshop on Open Source Software Engineering (ICSE 2002), Orlando, FL, USA, May 2002.

[10] GERMAN, D.M., GNOME, a case of open source global software development", Proc. of the Int. Workshop on Global Software Development (ICSE 2003), Portland, Oregon, USA, May 2003.

[11] GNOME.ORG, About The GNOME Foundation, http://foundation.gnome.org/about/

[12] GNOME.ORG, GNOME Bug Days, http://developer.gnome.org/projects/bugsquad/triage/faq.html#II

[13] GONG, L., JXTA: A network programming environment, IEEE Internet Computing, 5(3):88-95, May-June 2001.

[14] GOTTESDIENER, E., Facilitated Workshops in Software Development Projects, Int. Conference on Application of Software Measurement (ASM2001), San Diego, CA, USA, February 2001.

[15] HAYES, B., Crossing Oceans and Firewalls, Proc. of the Project Management Institute Global Congress North America., Baltimore, Maryland, USA, 2003

[16] HERBSLEB, J.D, GRINTER, R.E., Architecture, Coordination, and Distance: Conway's Law and Beyond, IEEE Software, 16(5):63-70, September/October 2001.

[17] HERBSLEB, J.D, MOITRA D., Global Software Development, IEEE Software, 18(2):16-20, March/April 2001.

[18] HERBSLEB, J.D, MOCKUS, A., FINHOLT, T.A., GRINTER, R.E., An empirical study of global software development: Distance and speed, Proc. of Int. Conference on Software Engineering (ICSE 2001), Toronto, Canada, May 2001.

[19] JARVENPAA, S., LEIDNER, D.E., Communication and Trust in Global Virtual Teams, Journal of Computer-Mediated Communication, 3(4), June 1998.

[20] JXTA.ORG, JXTA Bug Days, http://platform.jxta.org/servlets/ReadMsg?msgId=88911&listName=dev

[21] JXTA.ORG, IETF standardization effort, http://www.jxta.org/IETFStandard.html

[22] JXTA.ORG, Project JXTA Home Page, http://www.jxta.org

[23] JXTA.ORG, P2PConference Home Page, http://p2pconference.jxta.org

[24] JXTA.ORG, CMS Home Page, http://cms.jxta.org

[25] JXTA.ORG, Jxtacast Home Page, http://jxtacast.jxta.org

[26] LANUBILE, F., MALLARDO, T., Tool Support for Distributed Inspection, Proc. of International Computer Software and Applications Conference (COMPSAC 2002), Oxford, UK, 2002.

[27] LINUX.ORG, Linux-Kernel Archive, http://www.uwsg.iu.edu/hypermail/linux/kernel/0202.0/0989.html

[28] MILOJICIC, D.S. et al. Peer-to-Peer Computing, HP Laboratories Palo Alto, March 2002.

[29] MOCKUS, A., FIELDING, R.T., HERBSLEB, J.D, Two Case Studies of Open Source Software Development: Apache and Mozilla, ACM Transactions on Software Engineering and Methodology, 11(3):309-346, July 2002.

[30] PALMER, J.W., SPEIER, C., Teams: Virtualness and Media Choice, Proc. of the Hawaii Int. Conference on System Sciences (HICSS 1998), Kohala Coast, HI, January 1998.

[31] SUN.COM, Java Web Start Technology, http://java.sun.com/products/javawebstart.

[32] WEBEX.COM, WebEx web site, http://www.webex.com