A P2P Toolset for Distributed Requirements Elicitation

Filippo Lanubile
Dipartimento di Informatica, University of Bari
Via Orabona 4, 7026 Bari, Italy
lanubile@di.uniba.it

Abstract

Geographically distributed teams need nowadays models and tools to support a load of activities that usually take place through the direct interaction among people. Our research effort is aimed to understand how decentralized systems, based on a peer-to-peer architecture, can be exploited to support the key activities of global software development. As a first step, we have focused on requirements elicitation because it is among the most communication-rich processes of software development. This paper presents a toolset for distributed requirements elicitation, which is developed on the basis of a peer-to-peer infrastructure platform, called Groove.

1. Introduction

Globally distributed workgroups usually rely on centralized systems, mostly built on top of web-based development platforms. Two examples of centralized infrastructure are SourceCast [10] and SourceForge [11], two collaborative development platforms, that include web-based access to defect and issue tracking, version control and configuration management, mailing lists and discussion forums.

Our research effort is aimed to understand how decentralized systems, based on a peer-to-peer (P2P) architecture, can be exploited to support collaboration across time and space for global software development.

Collaborative P2P applications are increasingly becoming popular to exchange instant messages, share common information and applications, and jointly review and edit documents. Collaborative P2P systems exhibit the following advantages with respect to client-server counterparts: autonomy, intermittency, and immediacy.

- 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.

Under these conditions, a P2P collaborative infrastructure can complement or even replace client-server platforms for the creation of ad-hoc or small software teams. 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.

Among the many collaborative software engineering activities, the focus in this paper is directed at the software requirements activities, specifically requirements elicitation. Eliciting requirements engage different stakeholders, both from the customer and the developer sides, who need to intensively communicate and collaborate. As a key part of the requirements engineering process, requirements elicitation has a great impact on the later development activities; any omission and incompleteness may lead to important mismatches between customers needs and released product.

This paper introduces a distributed requirements elicitation toolset which was developed on the basis of a P2P infrastructure platform, called Groove. Section 2 presents the functionality of the collaborative toolset for eliciting requirements from geographically dispersed stakeholders. Section 3 describes the P2P platform which we used as an application framework. Section 4 concludes the paper and suggests opportunities for future joint research.

2. Requirements elicitation toolset

Requirements engineering is the most communication rich process of software development, and then the effectiveness of a requirements engineering process is greatly constrained by the geographical distance between stakeholders [3], as in the case of global software development.

Some simple P2P tools have been already introduced to support communication between distant stakeholders. For instance, Microsoft's NetMeeting has been used in requirements negotiation [2] for its instant messaging capabilities, to create a video and audio link between people, and to share desktop applications. But the features offered by generic tools provide a partial support to specific requirements engineering activities, and often there is no way to integrate them with commercially available requirements engineering tools, such as Rational RequisitePro.

For this reason, there is a need to develop a tool infrastructure to support globally distributed workgroups when developing requirements. Given the characteristics of autonomy, intermittency and immediacy, we believe that a P2P-based toolset is suited for the cooperation-intensive needs of requirements engineering activities.

As a first step, we focused on requirements elicitation which is regarded as the first activity in the requirements engineering process [4]. Eliciting requirements is an information gathering activity with the goal to identify system stakeholders, their needs and expectations, system objectives and boundaries. Elicitation techniques include questionnaires and surveys, interviews and workshops, documentation analysis and participant observation.

We developed a P2P toolset to be used for eliciting requirements in a distributed context. In the following we briefly describe each of the five tools comprising the integrated toolset.

2.1. Stakeholders tool

The first step in requirements elicitation is often the identification of who brings an interest in the software project, both on the customer and the supplier sides, and then will have an influence on requirements definition.

The *Stakeholders* tool acts as an archive of information regarding the stakeholders involved in a project. The tool can be thought as a shared contacts list augmented with the annotation of the role (even multiple roles) in the software project. Given the great variability for role definition, the tool was developed with a set of default roles from the Volere requirements specification template [9], although it can be customized according to different needs. The tool accounts for changes of roles during the project lifetime, and for multiple roles of a single stakeholder. Provided that a stakeholder has been included, he or she can communicate with the other ones by either sending asynchronous messages or by inviting to join a chat.

2.2. Interview tool

Interviews are frequently used in requirements elicitation to gather detailed information from stakeholders. It also allows the requirements engineer to hear different viewpoints that might need to be reconciled.

In this tool an interview is set up through a wizard. There are two types of available interviews: structured or unstructured; in the former case the requirements engineer can decide whether to prepare the interview from scratch or not. If not, the tool supplies a set of interview templates for the elicitation of high-level system goals and architecturally significant requirements [5]. Using these templates, it is also possible to trace questions with respect to a requirements taxonomy.

Another feature of this tool is the availability of multiple views of the interview according to the interviewee's role. In this way the same templates can be customized for different stakeholders.

2.3. Requirements tool

The *Requirements* tool is used as a shared repository of elicited requirements. The tool exploits the P2P feature of data sharing: data are replicated (and encrypted) on everyone's computer and made accessible even offline.

A requirement is described by the following attributes: ID, version, name, rationale, description, priority, difficulty, stability, and status. An additional attribute, requirement type, was added accordingly to the requirements taxonomy used in the interview templates. In this way it is possible to trace stored requirements with related questions from the *Interview* tool. Users can also define new requirements categories.

2.4. Workshop tool

The *Workshop* tool replaces the usual face-to-face workshop in which a group of experts and stakeholders

works together with the goal of information discovery and creation. Live discussions can be conducted either through a chat or an audio link.

A workshop requires an agenda to let participants follow a discussion flow, and two specific roles: facilitator and scribe. The facilitator leads the process by monitoring the discussion and managing group's dynamics. The scribe documents the group's work by taking notes as the discussion proceeds.

The tool helps to prepare the workshop through a wizard which lets to define the target, the discussion topics (they can be derived from interview logs), the participants (and their role) and the meeting schedule.

2.5. Vote tool

The *Vote* tool can be used in the context of a workshop, if controversies arise during discussion, or as a standalone tool as well.

Voting includes a preparation phase in which it is possible to add open or closed-answer questions to a shared list. Voting can be set up as a secret or evident ballot, depending whether or not participants' choices have to be kept anonymous.

3. Decentralized platform

We have implemented the requirements elicitation toolset on the basis of Groove, an extensible decentralized platform intended for communication, content sharing, and collaboration [6].

Collaboration activities with Groove take place in a shared application space, which is accessed from a rich application client, called *transceiver*. A shared space, including tools and persistent data, is duplicated on every space member's computer. Data within a shared space are encrypted, both on disk and over the network, to assure confidentiality and integrity. Both data and commands are transformed, stored and transmitted as XML documents.

Every modification made to a shared space is propagated to the other peers. If users do not stay connected at the same time, the shared space gets synchronized when a peer goes online. In this way the state of the shared space remains the same for all peers.

As other P2P systems, such as Napster, the actual interaction model of Groove is hybrid because peers can establish direct connections using a Groove's native P2P protocol called SSTP (Simple Symmetrical Transmission Protocol), while the following services are supplied through central servers:

- Presence awareness: when a peer running Groove goes online, it registers with a presence server. In this way other peers can scan the presence server if interested in collaborating with other ones.
- Relay: when a peer is offline, communications destined to it are sent to a relay server which will deliver data to the peer when it reconnects. Relay servers are also used when peers reside behind a firewall that only allows outbound connections. Since firewalls are usually configured to allow employees to access the web, Groove leverages this existing configuration to send and receive messages through HTTP tunneling.
- Fanout: when the same information must be sent to many users at the same time, data are transmitted once to a server which retransmits them to other peers.

With Groove, shared space members can use predefined tools supplied by Groove Networks or by third parties. Predefined tools include instant messaging, chat, threaded discussions, audio-conferencing, shared files, shared contacts, group calendaring, group editing, group drawing and co-browsing. Developers can also build their own applications in the form of single tools or integrated toolsets.

To build the distributed requirements elicitation toolset we used Groove as an application framework, by reusing Groove's critical services (storage, transport, encryption, synchronization, messaging, presence awareness) and default tools (e.g., Contact Manager, Outliner, Chat) with the addition of scripts and XML code, accordingly to the Groove Development Kit (at the time of development we used GDK 1.3).

4. Conclusions

P2P is not a new technology but it is now emerging as an alternative or complement to client-server models for designing collaborative development systems. In order to investigate issues that can be encountered when providing P2P tool support for global software development, we developed an integrated toolset for distributed requirements elicitation.

In the field of collaborative software development environments the P2P technology has begun to being introduced. At University of Calgary, a project has recently started [1] to port an existing process-support environment (called MILOS) from a client-server model to a P2P model, thus relaxing centralized control. The migration project intends to use JXTA [7] as a P2P framework., JXTA is a set of protocols for P2P applications without restrictions to particular operating systems or network services. Although the JXTA specification is open to any programming language, the most widely used implementation is in Java.

As for Groove, JXTA uses XML to exchange data between peers. However, JXTA and Groove are not fully comparable: the former is a low-level, general-purpose

platform and, hence, provides only services and no built-in tools, while the latter is a very full-featured collaborative application, which can also be used as a development platform. By the way, the author is taking part to the development of a P2P remote-conferencing tool, based on the JXTA framework. This is an open source project, hosted by jxta.org [8] and owned by Fabio Calefato at University of Bari.

We wish that this paper will rise a discussion on developing and using tools to support collaborative activities for global software development. We also hope that the workshop will give us the opportunity to start collaborations for experimenting with the requirements elicitation toolset.

5. References

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

[2] D. E. Damian, A. Eberlein, M. L. G. Shaw, and B. R. Gaines, "Using different communication media in requirements negotiation", *IEEE Software*, 17(3), May/June 2000, pp.28-36.

[3] D. E. Damian, "The study of requirements engineering in global software development: as challenging as important", *Proc. of the ICSE Int. Workshop on Global Software Development*, Orlando, FL, USA, May 2002.

[4] A.M. Davis, *Software requirements: analysis and specification*, Prentice-Hall Press, Upper Saddle River, NJ, 1990.

[5] P. Eeles, "Capturing Architectural Requirements", *The Rational Edge*, Nov. 2001, www.therationaledge.com/content/nov_01/t_architecturalRequirements_pe.html

[6] Groove Networks, http://www.groove.net

[7] jxta.org, *Project JXTA*, http://www.jxta.org

[8] jxta.org, *Project P2PConference*, http://p2pconference.jxta.org

[9] S. Robertson, and J. Robertson, *Mastering the Requirements Process*, Addison-Wesley, Boston, MA, 1999.

[10] CollabNet, http://www.collab.net

[11] SourceForge.net, http://sourceforge.net