Using Peer-to-Peer Technology to Support Global Software Development – Some Initial Thoughts

Seth Bowen
Department of Computer Science
University of Calgary
2500 University Drive N.W.
Calgary, Alberta, Canada T2N 1N4
bowen@cpsc.ucalgary.ca

Frank Maurer
Department of Computer Science
University of Calgary
2500 University Drive N.W.
Calgary, Alberta, Canada T2N 1N4
(403) 220-3531
maurer@cpsc.ucalgary.ca

ABSTRACT
Distributed software development typically uses a centralized architecture, which has some drawbacks such as, the participants may experience lengthy delays if they are located far from the central server, and the organization that runs the server must deal with the security and privacy issues that come with being in charge of a central repository of information. We are investigating whether this centralized control can be relaxed by using peer-to-peer (P2P) technology. Adopting a P2P architecture includes some of the following benefits for software development: (1) the peer (or group) is able to have complete control of its information, (2) groups can share and duplicate information to help users with slow network connections, and (3) users can easily contribute their own resources to the project, such as hard drive space. The P2P architecture also has potential drawbacks, including the need for complex search and retrieval algorithms, and having to coordinate the synchronization of duplicated stores of data. The objectives of our research are threefold: (1) to examine the design issues related to the development of a P2P software development application, (2) to alter an existing virtual software development application (MILOS) from a client-server to a P2P application using the Sun JXTA framework, and (3) to present empirical evidence on the value of the P2P implementation based on data gathered during the development process, and during application use (i.e., delays when searching and retrieving information).

Categories and Subject Descriptors
D.2.9 [**Software Engineering**]: Management – *programming teams*; C.2.4 [**Computer-Communication Networks**]: Distributed Systems – *distributed applications*

General Terms
Management

Keywords
Process Support, Peer-to-Peer Computing

1. INTRODUCTION
Globally distributed software development has led to the creation of several applications that are in widespread use today. Examples include telecommunication software from Eriksson, Nortel and others, and open source software, such as the Apache Web Server [1], the Linux operating system [7], the MySQL database [12], and the CVS (Concurrent Versions System) [2]. Open-source software is created typically in a centralized distributed topology, which involves people working in isolation and then releasing their source code to a central repository. Any coordination and communication that occurs between the members is done mainly through message bulletin boards and email. There are industry products available that developers can use over a network (e.g., Microsoft Project [10], Rational XDE [13]), and some tools developed in educational institutions (e.g., TUKAN [14] and CHIME [3]), however, most of these applications rely on a centralized architecture to aid the collaboration process. As a result, software developers participating in a distributed project (a) need to have a reliable and fast connection to the central server to access information and (b) need to trust the organization that runs the server.

In our research project, we are investigating if this centralized control can be relaxed by using peer-to-peer (P2P) technology. We see the following advantages compared to centralized approaches:

The reliability of the overall collaboration environment is increased because we replicate the information that is currently maintained on a central server. As a side effect of the replication, we also expect performance improvements because we can bring currently centralized information closer to each user.

Users have a tighter control over what information is accessible by other developers. This is especially necessary in virtual software enterprises, where the need for information sharing in a project needs to be balanced with the needs of participating companies on protecting their intellectual capital.

Our project is in its starting phase. The work presented here describes initial thoughts and should be seen as a means to start the discussions that could provide us with feedback on the overall idea.

The work presented in this abstract is based on our MILOS system - which is described in the next section. The last section

presents our current thinking on a P2P extension of the MILOS approach.

2. BACKGROUND

MILOS (Minimally Invasive Long-Term Organizational Support for software development) is an open-source project that provides support for the coordination and collaboration of virtual software engineering projects [8]. It includes the following developmental aids:

- a workflow engine supporting the management of projects, which includes the coordination of the tasks required for the completion of a project;
- a bug-tracking system;
- a metrics gathering utility; and,
- an experience base to store process models for future use.

MILOS also supports agile software development using some XP (eXtreme Programming) practices [9]. During the planning of a project, user stories can be created and edited. As well, the application supports the scheduling of communication and pair programming via Microsoft NetMeeting [11]. Although MILOS is geared towards software development, the application is general enough to support the development of any type of project.

3. ISSUES WITH MILOS AND P2P

MILOS is based on a client-server model that has some potential drawbacks for virtual software development teams. First, being dependent upon a central repository means that information may be brought together simply for the convenience of exchange or access. The potential problem is that the singular group in charge of this information must deal with the security and privacy issues related to this intellectual property, when several groups could perhaps more appropriately manage it. Second, a virtual environment often means that team members are distributed around the globe. The distance of a user to the central server can become a factor because team members may connect to the Internet via slow connections and so could be subjected to long delays.

The P2P networking model offers advantages in software development over a client-server model because of the independence from a central control. First, each peer, or group, is able to have complete control of its information, including source code. The group could even be more cohesive because they can choose what information to filter from the outside. Second, groups can dynamically share and duplicate information with one another to allow for faster access to users who may have slow network connections, or to support multiple projects that have dependencies on the same artifacts. Third, the developers can easily contribute their own resources, such as hard drive space and computing cycles, to an endeavor, which increases the robustness of the environment. This feature can be especially helpful for projects with limited funding, such as open-source projects.

There are also potential drawbacks that must be considered when adopting a P2P design. First, searching for information is in general more complicated and time-consuming in a P2P environment than in a client-server environment because the information is often more distributed, and can be more dynamic. These characteristics could lead to longer development times. Second, there is a greater importance on authenticating peers since the dynamic environment does not lend itself to easily tracing peers, such as through static IP addresses or registered domain names because these may not be used. When dealing with sensitive information, such as during the development of a medical information system, the authenticity of peers must be without doubt. Third, there is the issue of the coordination of information in a P2P environment. For example, if information is duplicated then how often should the stores be synchronized with one another?

4. OBJECTIVES

The objectives for our research of P2P virtual software development are threefold. First, to examine the design issues related to the development of a P2P virtual software development application, with specific attention given to the coordination of data between different peers, and security concerns. Second, to alter MILOS from a client-server to a P2P application using a P2P framework. And third, to assess the value of the P2P implementation based on comparisons of the two implementations based on the differences in the quality and quantity of information available, and performance. An empirical evaluation of the approach will involve gathering data on the development process, measuring performance by measuring the delays when retrieving information, and having users complete questionnaires on their impressions of how the P2P architecture impacts the original application. This research will present some novel findings into the usefulness and issues related to applying the P2P architecture to virtual software development.

5. IMPLEMENTATION FRAMEWORK

As MILOS is written in Java, we will be using the JXTA P2P framework for our extensions. The JXTA model is a general specification for describing a P2P architecture [4]. The specification is general in that it does not restrict the type of language, service, or even network that is used. For example, communication can be supported at the application layer by using HTTP, or even at the lower IP layer by using datagram packets. The JXTA specification is intentionally high-level to support the interoperability of any P2P system. Data is structured in JXTA using XML (eXtensible Markup Language), which is a common method of structuring data for the communication between different applications. Thus, any application can take advantage of the JXTA specification without having to be tied into a vendor. Open-source implementations of the JXTA specification are currently being developed in several languages, including Java, J2ME (Java Micro Edition), C, Perl, Python, Ruby, and Objective C [6]. There is currently a stable build available in Java because it was the first programming language used to implement JXTA. Java is an appropriate language for the implementation of a P2P specification because applications written in Java can run in several different operating systems, including Windows, Macintosh, Solaris, and Red Hat Linux [5].

6. REFERENCES

[1] Apache HTTP Server Project. http://httpd.apache.org/.

[2] Concurrent Versions System. http://www.cvshome.org/.

[3] Dossick, S.E., Port, D., and Kaiser, G.E. Embedding Model-Based Architecting in a Collaborative Environment. Columbia University, New York, NY. http://www.psl.cs.columbia.edu/ftp/psl/CUCS-016-99.pdf.

[4] Gong, L. JXTA: A Network Programming Environment. IEEE Internet Computing, 4 (May-June 2001), 88-95.

[5] J2SE v 1.8 Download. http://java.sun.com/j2se/1.8/download.html.

[6] JXTA Project. http://core.jxta.org/servlets/ProjectHome.

[7] Linux Online. http://www.linux.org.

[8] Maurer, F., Dellen, B., Bendeck, F., Goldmann, S., Holz, H., Kötting, B., and Schaaf, M. Merging Project Planning and Web-Enabled Dynamic Workflow Technologies. IEEE Internet Computing, 8 (May-June 2000), 65-74.

[9] Maurer, F. and Martel, S. Process Support for Distributed Extreme Programming Teams. University of Calgary, Calgary, Canada. http://sern.ucalgary.ca/~milos/papers/2002/MaurerMartel2002.pdf.

[10] Microsoft Project. http://www.microsoft.com/office/project/default.asp.

[11] Microsoft NetMeeting. http://www.microsoft.com/windows/netmeeting/.

[12] MySQL. http://www.mysql.com/.

[13] Rational XDE. http://www.rational.com/products/xde/index.jsp.

[14] Schümmer, T. and Schümmer, J. Support for Distributed Teams in eXtreme Programming. German National Research Center for Information Technology. http://www.darmstadt.gmd.de/concert/people/schuemmePrivate/xp00.pdf.