Embedding Social Networking Information into Jazz to Foster Group Awareness within Distributed Teams

Fabio Calefato, Domenico Gendarmi, Filippo Lanubile
Dipartimento di Informatica
Università degli Studi di Bari
via E. Orabona, 4 – 70125 – Italy
{calefato, gendarmi, lanubile}@di.uniba.it

ABSTRACT

A Collaborative Development Environments (CDE) provides a project workspace with a standardized toolset to help distributed development teams cope with geographical distance. However, there is a lack of support to reduce socio-cultural distance, which poses practical barriers to the development of connections and shared context/culture between team members. The rise of the Social Web has created several opportunities to publish personal information, often further composed through Web mashups, which can be regarded as a valuable data source in order to establish a shared context among remote developers, with little or no chances to meet.

In this paper we present our preliminary work that aims to provide distributed software teams with overall, contextual awareness aggregated in one place. Using the IBM Jazz as CDE, which already provides both presence and workspace awareness, we leveraged the FriendFeed aggregator service to embed personal information about distributed co-workers, collected from social networks. Disseminating additional group awareness information to developers, who have little or no chances to meet, can help to speed up the establishment of organizational values, attitudes, and trust-based inter-personal connections.

Categories and Subject Descriptors

D.2.6 [**Software Engineering**]: Programming Environments – Integrated environments, Interactive environments, Programmer workbench.

General Terms

Management, Human Factors, Design.

Keywords

Social Web, Web 2.0, Mashup, Jazz, Eclipse, Collaborative Development Environment, CDE, Group awareness.

1. INTRODUCTION

In distributed settings, due to distance, software teams often rely on Collaborative Development Environment (CDEs), such as SourceForge[1], GoogleCode[2], Github[3], and Assembla[4], which are an integrated and flexible set of tools that help distributed teams control their software development process. Yet, despite their ability to cope with geographical distance, CDEs provide little support to reduce socio-cultural distance, which poses practical barriers to the development of connections (i.e., common ground, mutual confidence, trust) and shared context/culture (i.e., assumptions, beliefs, attitudes, values) within distributed teams, with a potential severe impact on project management effectiveness. The idea of applying social software to help distributed teams deal with socio-cultural distance is rather recent. The rise of the Social Web, also known as Web 2.0 [3], created lots of sources for personal, user-generated content, which is often further composed through Web mashups. In [1], we argued that embedding into CDEs information collected from social websites (e.g., Twitter[5], Facebook[6], Last.fm[7]) might help to speed up the establishment of shared context and culture, as well as personal relationships between distant team members, with little or no chances to meet.

Ko et al. [8] conducted a study to understand the information needs in software developments teams, showing that the most frequently sought information included awareness about *tasks*, *artifacts*, and *co-workers*. However, recent research studies have also shown that tool support for distributed software development teams are still inadequate in enhancing distributed awareness because most tools are designed to answer a specific kind of "awareness questions" in isolation [11]. For instance, communication applications, such as IM and VoIP tools, provide *presence awareness* to help coworkers minimize interruptions and disturbances when engaging in collaborative processes. Instead, tools such as Palantìr [11], Hipikat [4], and Mylyn [7] provide developers with *workspace awareness*, which help to identify other developers, artifacts, and tasks that are related to the artifact/task at hand. IBM Jazz is one of the most recent and full-featured CDE, which provides both presence and workspace awareness in one place [5].

SoSEA'09, August 24, 2009, Amsterdam, The Netherlands.

[1] http://sourceforge.net/

[2] http://code.google.com

[3] http://github.com/

[4] http://www.assembla.com/

[5] http://twitter.com/

[6] http://www.facebook.com/

[7] http://www.lastfm.it/

However, neither Jazz supplies informal, personal information on coworkers in order to provide them with *group awareness*, that is, the consciousness of individual collaborators in a group with regards to affective behaviors, collective orientation, particularism or diffuseness [13], which is extremely beneficial to maintain a sense of teamness between geographically distributed software teams [10].

In this paper we present our initial work that aims to provide distributed software teams with overall, contextual awareness (i.e., presence, workspace, and group awareness) aggregated in one place. Using the Jazz CDE, which already provides both presence and workspace awareness, we leveraged the FriendFeed aggregator service to embed personal information from social networks about distributed co-workers, who have little or no chances to meet, in order to provide group awareness and, at the same time, speed up the establishment of organizational values, attitudes, and trust-based personal connections between them.

The remainder of this paper is organized as follows. In Section 2 we present an overview on CDEs and Jazz, in particular. In Section 3 we first discuss the rise of Social Web applications in general, and then we present FriendFeed as a particular example of Web Mashup. Our Jazz extension is discussed in detail in Section 4. Finally, in Section 5 we conclude.

2. COLLABORATIVE DEVELOPMENT ENVIRONMENTS

In software development, control is the process of adhering to goals, policies, standards, and quality levels, set either formally (e.g., formal meetings, plans, explicit guidelines) or informally (e.g., team culture, peer pressure). Because in distributed settings it is not possible to control units by walking, organizations had to fall back to using collaborative tools to control the software development process from a distance. Collaborative Development Environments (CDEs), such as SourceForge, Gforge, Google Code, are the most used and full-featured process-aware tools to support distributed teams.

CDEs were envisioned by Booch & Brown, who first acknowledged the need for 'frictionless surface' in development environments [2], motivated by the observation that much of the developers' effort is wasted in switching back and forth between different applications to communicate and work together. According to this vision, collaborative features should be available as components that extend core applications (e.g., the IDE), thus increasing the users' comfort and productivity. Therefore, CDEs support developers by incorporating the standard toolset needed (e.g., compiler, debugger, version control system, bug tracker) within a single project workspace, reducing the effort of running multiple different applications to communicate and work together.

Earliest CDE were developed within open source software (OSS) projects because OSS projects, from the beginning, have been composed of dispersed individuals. Today a number of CDEs are also available as commercial products.

2.1 IBM Jazz

Jazz [5] is one of the most noticeable commercial CDE because it can be customized to support any process. Besides, Jazz is an extensible platform, which leverages the Eclipse's notion of plugins to build CDE products. The present version has a wide-ranging scope, but in the former version of Jazz the goal was to integrate synchronous communication and reciprocal awareness of coding tasks into the Eclipse IDE, following Booch & Brown's vision.

Jazz is an extensible team collaboration platform based on a client-server architecture, which integrates many different technologies in a single environment. The Jazz server hosts a set of services (e.g., generate reports, resolve work items from the Web) and houses data in its repository (e.g., configurations, source code). Remote clients communicate with the Jazz server over the network, using SOAP/XML over HTTP (Figure 1). The full-featured Jazz client is Rational Team Concert, an extension of the Eclipse IDE, packed with all the plugins necessary to the Jazz development platform. It provides presence awareness, thanks to the integration with Lotus Sametime, and workspace awareness, by generating an RSS feed of all project-related events occurring the workspace.

The essential components of Jazz are the Repository and Team Process, which represent the platform kernel and are developed by the Jazz Project Member (Figure 2). Other members of the Jazz community develop additional components to add new capabilities to Jazz, such as source control and reporting. While the Team Process component is meant to make Jazz a customizable, process-aware platform, Repository allows to store tool-specific information in a central place where it can be made available to all other components in all client and server configurations. Thus, the Repository plays a key role in Jazz extensibility.

3. SOCIAL WEB & MASHUPS

Recently, the rise of the Social Web [3] , created new incentives and motivations for publishing personal information on the Web. Nowadays, plenty of user-generated content is public available. Applications like Wikipedia[8], Flickr[9], Delicious[10], LibraryThing[11], for example, provide each day new wiki articles, photos, bookmarks and book reviews, as well as new metadata, which are directly added by users.

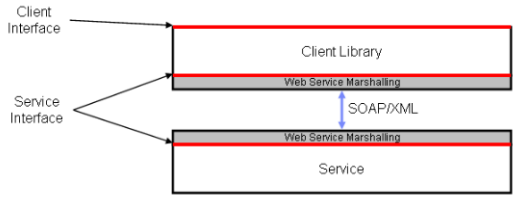

Figure 1. Jazz Client-Server Communication [6]

[8] http://www.wikipedia.org/

[9] http://www.flickr.com/

[10] http://delicious.com/

[11] http://www.librarything.com/

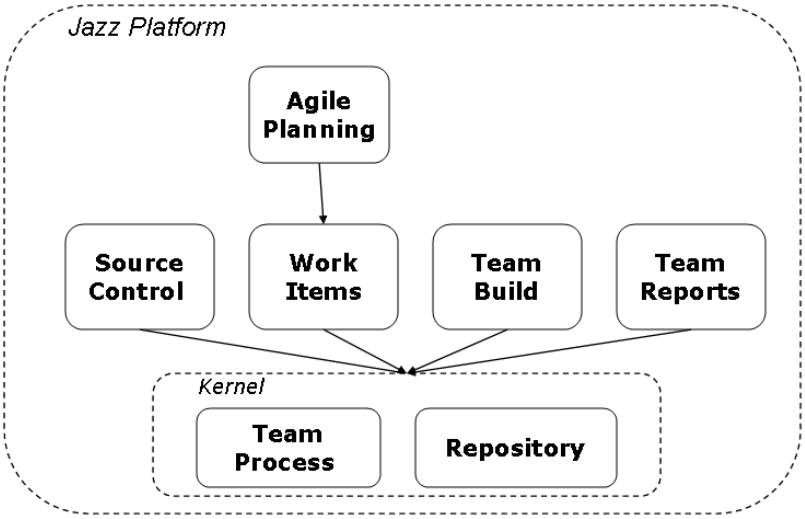

Figure 2. Components of Jazz Platform [6]

However, these forms of collaborative contributions are restricted to one single application and current Social Web applications are isolated from one another, like 'walled gardens'. The main reason for this lack of interoperation is that for the most part in the Social Web, applications' owners are quite reluctant to provide programmatic access to user generated content, which is hosted within their web sites.

In such a context, Web mashups have emerged providing a dynamic approach to compose content and functionalities originating from disparate web sources [14]. Among the different classes of mashup available on the Web, we focused on News Mashups [9] also known as Syndication Feed Mashups since they use syndication technologies like RSS and Atom to aggregate news related to various topics and create personalized feed views. In particular, we envisioned the opportunity to aggregate feeds about personal information originating from social networking sites in order to foster group awareness within distributed teams.

3.1 FriendFeed, a Social Information Aggregator

FriendFeed[12] is a real-time feed aggregator that consolidates the updates from a number of social networking websites (e.g., Delicious, YouTube[13], Last.fm, LinkedIn[14], Facebook), as well as any other custom website providing an RSS/Atom feed. Thus, FriendFeed users can use this stream of information to create customized feeds to share (and comment) with friends. The main reason of FriendFeed success is that it provides the facility to track users' activities (such as posting on blogs, Twitter and Flickr, or listening to music on Pandora[15]) across a broad range of different social networks, whereas other services exclusively facilitate tracking of their own members' activities on their particular social service.

What looked attractive to us about exploiting the FriendFeed service into Jazz is that: (1) a free API is available for leveraging the service output into third-party applications; (2) private groups can be defined so that updates from members are bound within group and not visible to users on the outside; (3) users can decide what is relevant or appropriate to stream and share, reducing information privacy and overloading concerns.

Members of the same Jazz project-area can create a FriendFeed private groups where they can choose which feeds they want to share and who can see the shared feeds. They can also start a conversation around shared items, or just show that they like a feed someone else has shared. By aggregating in the same place different feeds from social networking services used on a daily basis by the project members, we can thus foster the discovering and discussion of personal information regarding people within the project-area, speeding up the development of connections and shared context between team members.

4. THE JAZZ EXTENSION

The Jazz client extension was coded using a Java wrapper of the FriendFeed API. Unfortunately, the API itself does not allow doing as much as desired. For instance, groups cannot be created or managed using the API, which basically allows getting group description and members, reading the feed, and posting messages. In the remainder of this section, we briefly illustrate how the extension works in a simple scenario.

Using the FriendFeed web interface, the project-lead of a Jazz project-area creates a private group for the project. In order to avoid the friction of switching to the web browser and then back to the Jazz client, a browser window can be opened in one of the workspace views to complete group creation (see Figure 3a).

Other developers needs to be invited to join the group, using either their FriendFeed usernames, in case they are already

[12] http://friendfeed.com/

[13] http://www.youtube.com/

[14] http://www.linkedin.com/

[15] http://www.pandora.com/

registered, or just emails (see Figure 3b). Because the group is private, developers will have to enter their FriendFeed credentials to subscribe the aggregated feed and post contribution. Using the Eclipse Modeling Framework, we extended the project-area model in order to store FriendFeed credentials about both users and groups into the Jazz server repository. This way, even when changing the machine where the Jazz client is run (e.g., in case of switching from the office desktop to the laptop for a business travel), the group feed is available as long as the extension is installed on the client.

Developers can be invited to the group as regular members or admins: Only in the latter case they are allowed to add a service to the aggregated feed. As a group admin, each developer is allowed to specify what personal information collected from external social web sites can be streamed to the group, without violating his/her privacy. For instance one developer can choose to stream what he likes on Last.fm and Pandora, whereas another one can share her bookmarks saved in Delicious and the book reviews she posted on LibraryThing.

Once the group is created, the project-lead registers it with the project area in Jazz by entering the URL of the aggregated feed. Upon registering the group, the associated feed appears in the Team Artifact view of Jazz, under the Feed folder (see Figure 4a), along with the default feeds that provide workspace awareness by informing developers about development-related events (e.g., commits, build failures) or any other change occurring in the workspace (e.g., changes to the milestones release date). Hence, the aggregated feed of personal information is visualized using the same internal feed reader provided by Jazz (see Figure 4b). Because source services can be added to the aggregated stream by each developer who subscribed the FriendFeed group as admin, the larger the development team, the higher the risk of information overload is. Hence, we extended the feed reader to give each developer the opportunity to filter the updates from undesired sources. For instance, one can choose to filter out the updates from Facebook, YouTube and Flickr, while displaying all the others available in the aggregated feed (see Figure 4c).

Finally, an extra view has been made available in the workspace to let developers post message to the FriendFeed group directly from the Jazz client (see Figure 5a). All subscribers to the group are then able to view new messages as feeds, through both the FriendFeed web-based interface and the feed reader included within the Jazz client (Figure 5b).

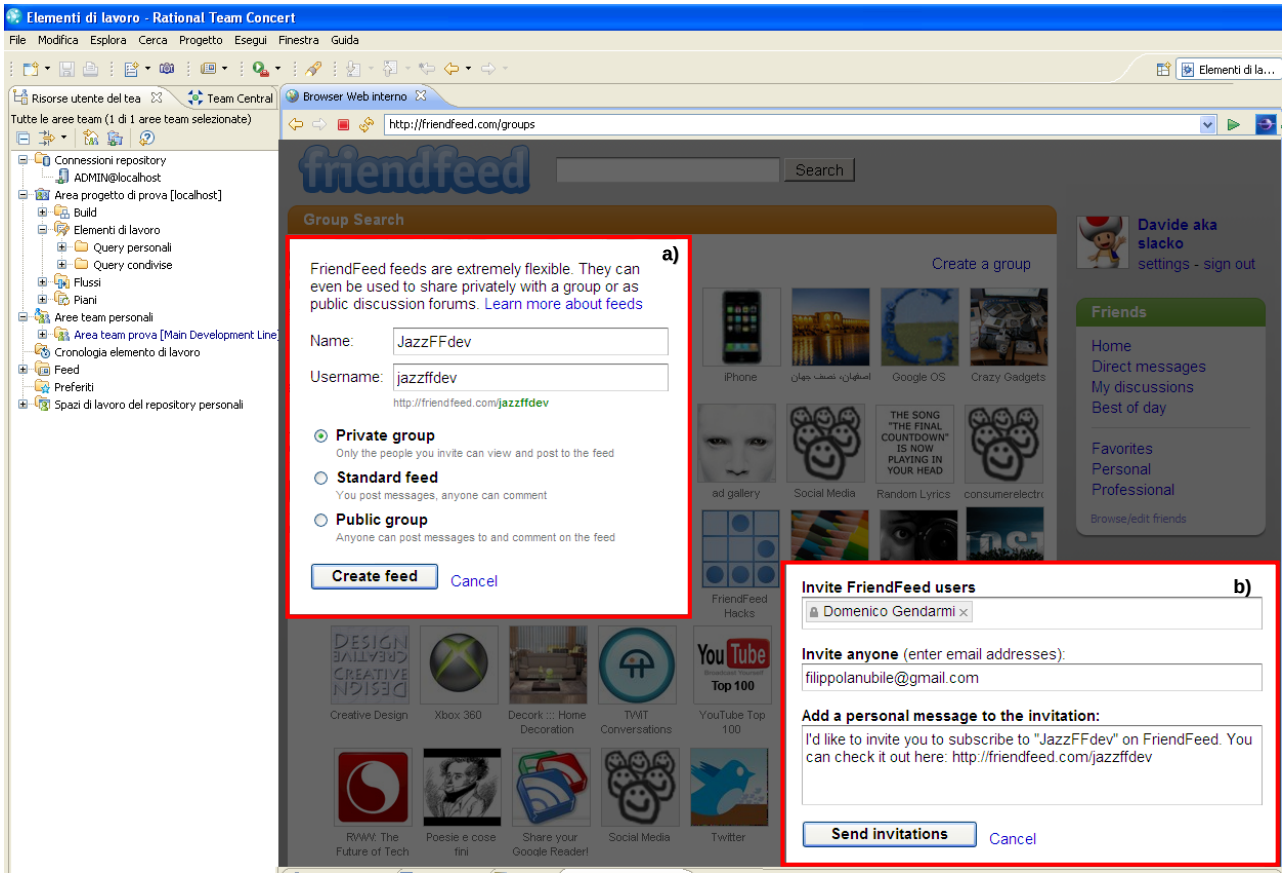

Figure 3. FriendFeed group creation *a)* Friend Invitation to the group *b)*

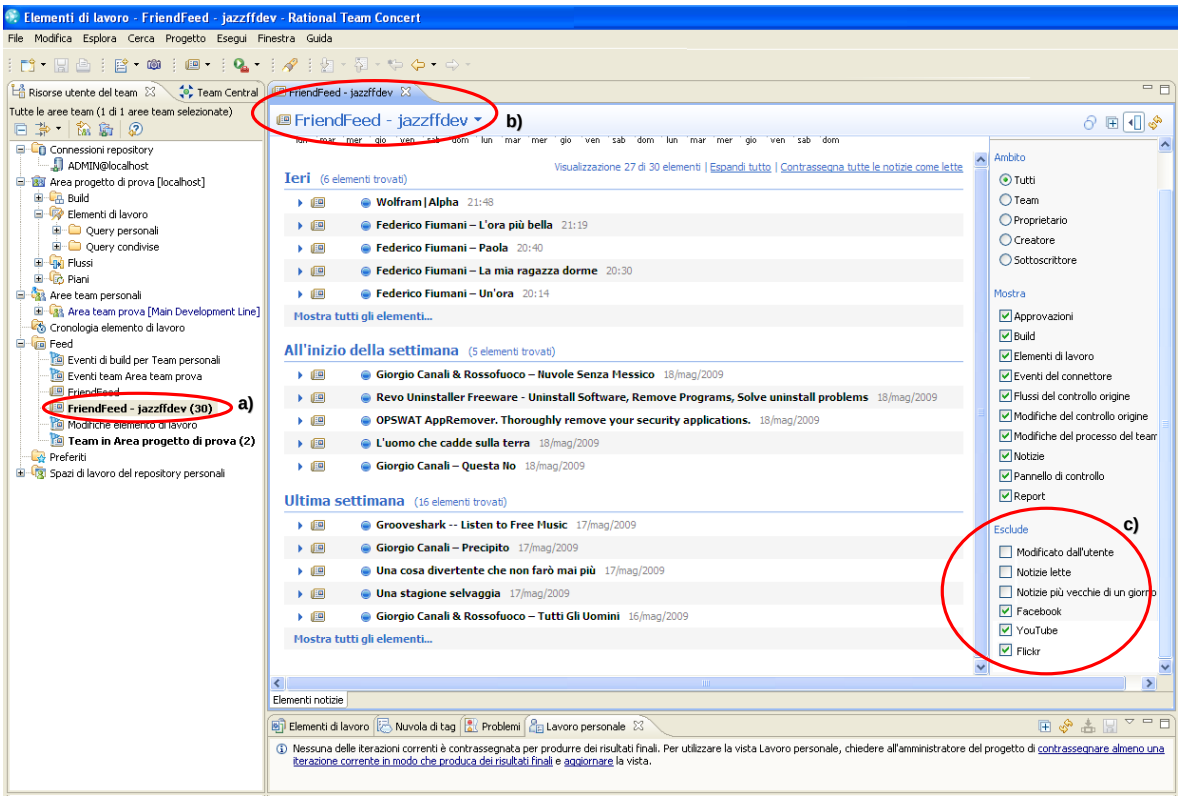

Figure 4. Name of the feed for the created group a) Feed view on Jazz b) Filtering options c)

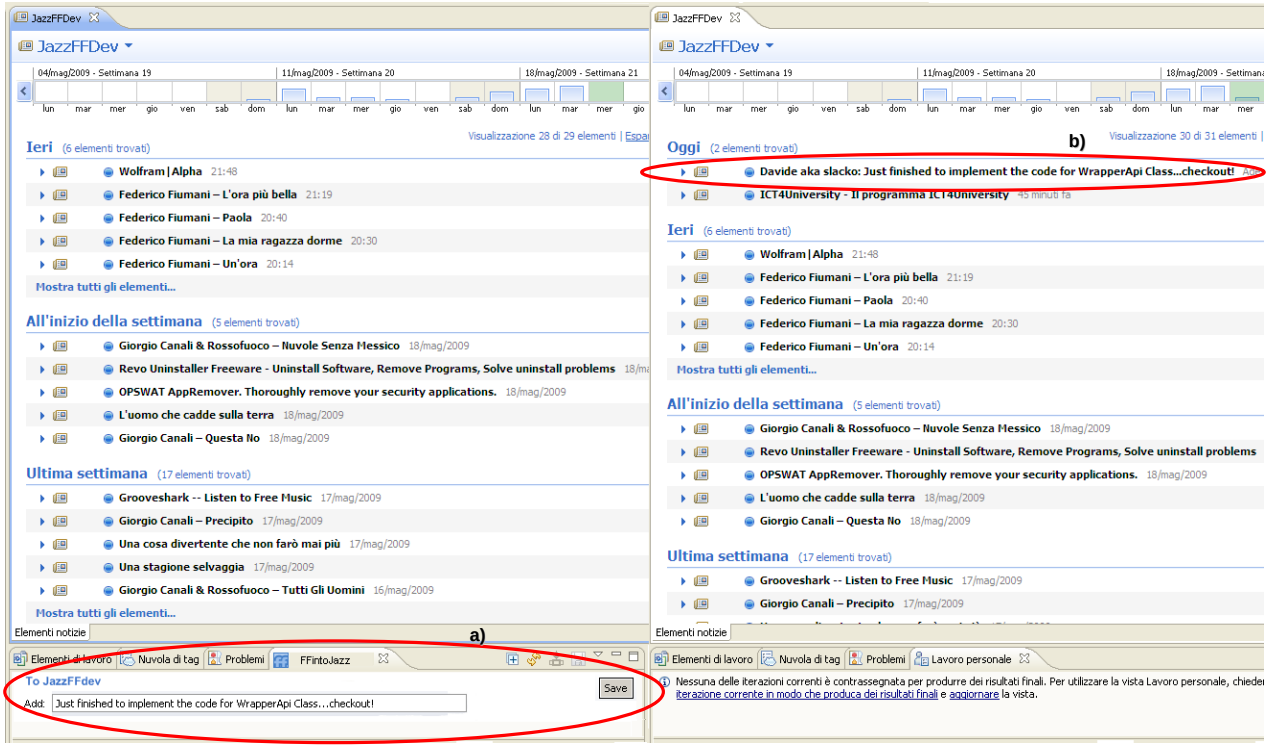

Figure 5. Posting a message to the group *a)* View of the updated feed *b)*

5. CONCLUSIONS & FUTURE WORK

Jazz is a Collaborative Development Environment that provides developers with both presence and workspace awareness. In this paper we have presented an extension of the Jazz platform, which leverages the feed aggregation service of FriendFeed to embed social information collected from social networks into the Jazz client and provide developers with group awareness as well. We argue that disseminating additional group awareness information to developers working in dispersed teams can help to speed up the establishment of organizational values, attitudes, and trust-based inter-personal connections, thus facilitating the overall distributed software development process.

While current work is focusing on the enrichment of the Jazz client, as future work we plan to extend Jazz also on the server side by augmenting the Jazz user profile with new professional information. Through the mashup of information collected from both business- and development-oriented social websites (e.g., LinkedIn and Ohloh), we intend to give to Jazz users the opportunity to customize their profile by including information about professional connections and expertise.

Extending Jazz server-side model allows to leverage collected information in the whole profile rather than within a specific project area, thus enabling information sharing within different teams inside Jazz. Moreover, we plan to develop the server-side mashup as a web service that expose personal information as RDF Linked Data in order to enable Jazz users exporting professional profiles and reusing them in third-party relevant applications.

6. ACKNOWLEDGMENTS

Our thanks to Davide Fucci for implementing the initial prototype of our Jazz client extension.

7. REFERENCES

[1] Abbattista, F., Calefato, F., Gendarmi D., and Lanubile, F. Incorporating Social Software into Agile Distributed Development Environments. Proc. 1st ASE Workshop on Social Sofware Engineering and Applications (SOSEA 2008), L'Aquila, Italy, 15 September 2008.

[2] Booch, G. and Brown, A.W., Collaborative Development Environments, Advances in Computers 59, 2003.

[3] Chi, E.H., The Social Web: Research and Opportunities, IEEE *Computer* , vol.41, no.9, pp.88-91, 2008.

[4] Cubranic, D., Murphy, G.C., Singer, J., and Booth, K.S. Hipikat: a project memory for software development. IEEE Transactions on Software Engineering, 31(6):446-465, 2005.

[5] Frost, R., (2007). Jazz and the Eclipse Way of Collaboration. IEEE Software, 24(6), pp. 114-117.

[6] Jazz Platform Technical Overview, https://jazz.net/learn/PrintableLearnItem.jsp?href=content/docs/platform-overview/index.html

[7] Kersten, M. Focusing knowledge work with task context. PhD Thesis, University of British Columbia, 2007.

[8] Ko, A.J., DeLine, R., and Venolia, G. Information Needs in Collocated Software Development Teams. Proc. 29th international conference on Software Engineering, Minneapolis, 2007.

[9] Merrill, D. 2006. Mashups: The new breed of Web app. An introduction to mashups. IBM developerWorks. http://www.ibm.com/developerworks/xml/library/x-mashups.html

[10] Omoronyia, I. Sharing awareness during distributed collaborative software development. PhD Thesis, University of Strathclyde, November 2008.

[11] Sarma, A., Noroozi, Z., and Hoek, A. Palantír: Raising Awareness among Configuration Management Workspaces. Proc. 25th Int'l Conf. on Software Eng. Portland, 2003.

[12] Sillito, J., Murphy G.C., and De Volder, K. Asking and Answering Questions during a Programming Change Task. IEEE Trans. on Software Engineering, 34(4):434-451, 2008.

[13] Totter, A., Gross T., and Stary, C. Functional versus Conscious Awareness in CSCW-Systems. XV. IFIP World Computer Congress. Telecooperation - The Global Office, Teleworking and Communication Tools-Vienna, 1998.

[14] Yu, J., Benatallah, B., Casati, F., and Daniel, F. 2008. Understanding Mashup Development. IEEE Internet Computing 12, 5, 44-52.