Incorporating Social Software into Agile Distributed Development Environments

Fabio Abbattista, Fabio Calefato, Domenico Gendarmi, Filippo Lanubile
Università degli Studi di Bari, Dipartimento di Informatica
{fabio, calefato, gendarmi, lanubile}@di.uniba.it

Abstract

Collaborative development tools have become mainstream technologies for distributed teams, but they fall short when both agility and distance occur at the same time. Recently the use of social software applications, such as wikis and blogs, have emerged as a practical and economical option to consider as global teams may use them to organize, track, and publish their work and then, share knowledge.

We intend to push further the application of social software principles and technologies into collaborative development environments for agile and distributed projects. As a first step, in this paper we first present a survey of social software, as well as tools and environments for collaborative development. Then, we present some opportunities and challenges of incorporating social software aspects in agile distributed development.

1. Introduction

Distributed software teams are characterized by geographical, temporal and sociocultural distance, each having an impact on all the forms of cooperation within teams, that is, communication, collaboration and coordination [1],[4]. While distance exacerbates the importance of human-centered aspects related to collaboration in distributed development, even further challenges arise when teams apply agile development methods. Agile methods are based upon intense interactions among individuals and thus, they emphasize the need for frequent informal communication and collocated teams [10].

Previous research has already acknowledged that conventional agile methods need to be adjusted in globally distributed environments. In fact, although a number of strategies [14] for supporting distributed development have been proposed, little is known about how to address new challenges coming up when agility is introduced. Recent studies in literature have tried to address this challenge, providing recommendations for achieving both agility and distribution, focusing either on how to improve informal communication and agreement in agile distributed development scenario [12],[11], or on strategies and practices to enhance both flexibility and rigor, thus finding the right balance between agile and distributed approaches [13],[18].

Recently, besides classic groupware tools (e.g., email, shared calendars), other collaborative applications, known as social software (e.g, blog, wiki), have shown an appealing ability to overcome typical issues of remote group interaction, by making easier to communicate, collaborate, and share knowledge. With this paper we begin to investigate how incorporating social software principles and technologies into collaborative development environments can also improve communication and knowledge sharing for distributed teams applying agile methods.

The remainder of this paper is structured as follows. Section 2 introduces fundamental principles and characteristic functionality of the most popular social software applications. Section 3 presents an overview of software engineering tools for flexible and distributed development. Finally, Section 4 ends up with a discussion on opportunities and challenges of integrating social software aspects in agile and distributed development environments.

2. Social software

Social Software is a general term encompassing a set of tools and applications that enable group interaction and computer-mediated communication. The same concept is sometimes expressed as Web 2.0 [17] as it captures distinctive features and key principles that can be reduced to:

- *Participation*, content is created and organized by common users rather than organizations;
- *Interaction*, applications available from the Web provide rich user interfaces as desktop applications;

- *Community/collaboration*, the more people use such tools, the better they get, and social network effects emerge.

In the following, we report some key applications of social software.

Blogs. They represent the simplest way for common users to create a website where content is added in form of posts displayed in a reverse chronological order. Compared to traditional personal websites, the novelty stands in the opportunity for readers to interactively leave comments that interconnect different blogs to each other, generating a particular community or social network of bloggers, called blogosphere. Among the very many existing uses, multi-author blogs can be adopted within a software team for fostering communication among developers and customers, as a process document, and as a stepwise history of project evolution.

Wikis. They can be regarded as a simple web-based collaborative authoring system. Unlike blogs, a wiki may not distinguish among reader and writer, and let anyone create and edit content in form of wiki pages. Main wiki features include versioning and document history, which let users track changes applied to documents. As shown in [15], a wiki can be successfully used in the software development process, because it results as an excellent mean to collect asynchronous contributions from a group of distributed people in a centralized repository of textual artifacts.

Collaborative tagging and folksonomies. The activity of labeling resources of interest is called tagging, as it consists of attaching one or more tags (i.e., free keywords) to the resource. While individually using a tagging system, everyone can see who else is participating by observing others' tagging behaviors. This tight feedback loop makes these systems social and the result is a collection of annotations, also called a folksonomy [9]. Nowadays, different kind of systems have introduced folksonomy-based approaches for information organization, however social bookmarking systems like del.icio.us were the pioneers and made tags so popular, letting users store, organize (through tags), and share bookmarks to digital artifacts.

Social networking. Social networking sites support users in the shaping of a digital identity through the creation of profiles and networks of contacts [5]. Popular examples include both professional networks, like Facebook or Linkedin, and unprofessional ones, such as MySpace. The creation and maintenance of a personal publicly accessible profile plays a crucial role as it serves as a testimonial of digital self-presentation, not only for your existing friends, but also for making new ones. This is an important incentive for users to keep their profiles up to date and enlarge their networks.

Other Social Software. There are other numerous applications that can be regarded as social software. Mashup services, for example, combine data and services from different sites into a single access point, providing new functionality. By opening up data and quickly combining different information in new interesting ways, mashups can effectively exploit the large amount of user-generated content available through social software applications.

3. Tool support for collaborative software development

Tools provide a considerable help to software engineering activities. In the case of global software engineering, adequate tool support is paramount to enable distributed teamwork. Software engineering tools to assist distributed projects may fall into the following categories:

Software Configuration management. A software configuration management (SCM) tool includes the ability to manage change in a controlled manner, by checking components in and out of a repository, and the evolution of software products, by storing multiple versions of components, and producing specified versions on command. SCM tools also provide a good way for programmers to share software, making sure that interdependent files are changed together and controlling who is allowed to. Further SCM tools make it possible to save messages about what changed and why. Open-source SCM tools, such as SVN and its predecessor CVS, have become indispensable tools for coordinating the interaction of distributed developers.

Bug and issue tracking. This function is centered on a database, accessible by all team members through a web-based interface. Other than an identifier and a description, a recorded bug includes information about who found it, the steps to reproduce it, who has been assigned on it, which releases the bug exists in and it has been fixed in. Trackers are a generalization of bug tracking systems to include the management of other issues, such as feature requests, support requests, or patches.

Build and release management. It allows projects to create and schedule, typically through a web interface, workflows that execute build scripts, compile binaries, invoke test frameworks, deploy to production systems, and send email notifications to developers. The larger the project, the greater the need for automating the build and release functions. Build tools, such as CruiseControl and its ancestor, the UNIX make utility, are essential tools to perform continuous integration, an agile development practice that allows developers to integrate daily, thus reducing integration problems.

Knowledge center. This function is mostly document-driven and web-enabled, and allows team members to share explicit knowledge across a work unit. A knowledge center includes technical references, standards, frequently asked questions (FAQs) and best practices.

Communication tools. Software engineers have adopted a wide range of synchronous and asynchronous communication technologies for project use in addition or replacement of communicating face-to-face by speech. Email is the most-widely used and successful collaborative application. Thanks to its flexibility and ease of use, email can support conversations, but also operate as a task/contact manager. However, one of the drawbacks of email is that, due to its success, people tend to use email for a variety of purposes and often in a quasi-synchronous manner. In addition, email is 'socially blind' [7], because it does not enable users to signal their availability. Recently, chat and IM have been spreading more and more in the workplace because, unlike email, they are 'socially translucent', providing a lightweight means to ascertain availability of remote team members and contact them in a timely manner.

Collaborative development environments (CDE). They were envisioned by Booch and Brown, who first acknowledged the need for 'frictionless surface' in development environments [3], 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, a CDE provides a project workspace with a standardized toolset to be used by the global software team. 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 or research prototypes to enable distributed software development.

SourceForge, from CollabNet, is the most popular CDE with over 170.000 hosted projects and over 1.800.000 registered users. Its original mission was to enrich the open source community by providing a centralized place for developers to control and manage OSS projects development. SourceForge offers a variety of free services: web interface for project administration, trackers, mailing lists and discussion forums, download notification of new releases, shell functions and compile farm, and CVS- and SVN-based configuration management. The commercial versions for corporate use add features for tracking, measuring and reporting on software project activities. GForge is a fork of SourceForge ver. 2.61. It has been downloaded and configured as in-house server by many industrial and academic organizations. Like SourceForge, it also offers a commercial version, called GForge Advanced Server. Finally, another large index of free open source software is Freshmeat.

Trac provides an integrated wiki, an issue tracking system and a front-end interface to SCM tools, usually SVN. Project overview and progress tracking is allowed by setting a roadmap of milestones, which include a set of so-called "tickets" (i.e., tasks, feature requests, bug reports and support issues), and by viewing the timeline of changes. Trac also allows team members to be notified about project events and ticket changes through email messages and RSS feeds.

MASE is a CDE developed at University of Calgary [16]. MASE puts a great emphasis on knowledge sharing as it integrates both informal and formal knowledge representations into a wiki-based experience repository.

Jazz is an extensible platform, which leverages the Eclipse's notion of plug-ins to build CDE products [8]. 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 and Brown's vision [3].

4. Discussion

Agile and distributed development practices are so different that, when blended together, the key characteristics of the former exacerbate the challenges intrinsic to the latter. For instance, the need for regular communication, flexible requirements, and informal agreement of agile development contrasts with the large intrinsic reduction of communication, as well as the need for stable requirements and controlled processes typical of distributed software development.

However, although previous studies already suggested balancing practices for tackling these new dichotomous challenges [18], social software is now emerging as a practical and economical option to consider for applying the following practices:

Improve communication. Wikis and blogs are particularly valuable in distributed projects as global teams may use them to organize, track, and publish their work [15]. Recently the use of these collaborative web-publishing applications has become quite common, especially for OSS software projects. In the case of agile distributed development, these forms of social software can increase the amount of informal communication exchanged between team members.

Facilitate knowledge sharing. TagSEA [19] is a research effort that has explored the usefulness of letting developers annotate pieces of source code with free tags. In agile distributed settings the maintenance of the whole process and product repositories can benefit from the existence of tags used to annotate also artifacts other than pieces of source code, bringing in personal and collective rewards.

Build team trust and culture. The necessity to reduce project costs often makes unfeasible to organize frequent visits by distributed partners, in order to build team trust and culture. Traditional global software development already involves dynamic and evolving communication-based social networks, often mined for investigating collaboration and awareness patterns [6]. Agile distributed teams could also exploit social network profiles as a mechanism to develop digital identities, establish connections, and thus, build trust and common culture among people working on same projects.

Nevertheless, although the use of social software can be beneficial for agile distributed software development, new challenges raise from the observation that successful examples of social software adoption rely on both large and non-compulsory users participation. Whether agile and distributed development environments could meet the same condition is a research question that needs further investigation.

Fun factor. Making the use of social software mandatory for agile teams may take away from developers the fun of using such tools, a factor that usually fosters their adoption in general purpose scenarios [2].

Critical mass. Social software has proven useful only after building a large base of thousands users or more. Instead, in the context of distributed software development, the overall number of developers scattered over the remote sites is notable lower.

Collaboration as side effect. The social side of popular social software applications usually does not represent the main incentive to individual participation but it is rather an emerging side effect. People contribute mostly for a private motivation, however the global scale of the Web, at the end, brings additional collective benefits. Does the same principle hold for different contexts, such as agile distributed software teams?

References

[1] Ågerfalk, P.J., and Fitzgerald, B., "Flexible and Distributed Software Processes: Old Petunias in New Bowls?", *CACM*, 49, 10, pp. 27-34, 2006.

[2] Avram, G., "At the crossroads of knowledge management and social software", *Electronic Journal of Knowledge Management*, 4, 1, pp. 1-10, 2006.

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

[4] Carmel E., *Global Software Teams: Collaborating Across Borders and Time Zones*, Prentice Hall PTR, San Francisco, CA, 1999.

[5] Churchill, E.F. and Halverson, C.A. "Guest Editors' Introduction: Social Networks and Social Networking", *IEEE Internet Computing*, 9, 5, pp.14-19, 2005.

[6] Damian, D., Izquierdo, L., Singer, J., and Kwan, I., "Awareness in the Wild: Why Communication Breakdowns Occur", Proc. of the Int. Conf. on Global Software Engineering, pp. 81-90, 2007.

[7] Erickson, T., and Kellogg, W.A., "Social Translucence: An Approach to Designing Systems that Support Social Processes", *ACM Transactions on Computer-Human Interaction*, 7, 1, pp. 59-83, 2000.

[8] Frost, R., "Jazz and the Eclipse Way of Collaboration", *IEEE Software*, 24, 6, pp. 114-117, 2007.

[9] Golder, S., and Huberman, B., "Usage patterns of collaborative tagging systems", *Journal of Information Science*, 32, 2, 2006.

[10] Highsmith, J., and Cockburn, A., "Agile Software Development: The Business of Innovation", *Computer*, 34, 9, pp. 120-122, 2001.

[11] Korkala, M., and Abrahamsson, P., "Communication in Distributed Agile Development: A Case Study", EUROMICRO SEAA Conference, 2007.

[12] Layman, L., Williams, L., Damian D., and Bures, H., "Essential communication practices for Extreme Programming in a global software development team", *Information and Software Tech*, 48, 9, pp. 781-794, 2006.

[13] Lee, G., DeLone, W., and Espinosa, J. A., "Ambidextrous coping strategies in globally distributed software development projects", *CACM*, 49, 10, pp. 35-40, 2006.

[14] Lings, B., Lundell, B., Agerfalk, P.J. and Fitzgerald, B. (2007) A reference model for successful Distributed Development of Software Systems, In International Conference on Global Software Engineering (ICGSE 2007), Munich, Germany August 27-30, 2007, pp. 130-139

[15] Louridas, P., "Using Wikis in Software Development", *IEEE Software*, 23, 2, pp. 88-91, 2006.

[16] Maurer, F., "Supporting Distributed Extreme Programming", XP Agile Universe, Extreme Programming and Agile Methods, LNCS Vol. 2418, pp. 13-23, 2002.

[17] Murugesan, S., "Understanding Web 2.0", *IT Professional*, 9, 4, pp. 34-41, 2007.

[18] Ramesh, B., Cao, L., Mohan, K., and Xu, P., "Can distributed software development be agile?", *CACM*, 49, 10, pp. 41-46, 2006.

[19] Storey, M.-A., Cheng, L.-T., Bull, I. and Rigby, P. "Shared Waypoints and Social Tagging to Support Collaboration in Software Development", CSCW, pp. 195-198, 2006.