Editor: Christof Ebert ■ Vector Consulting ■ christof.ebert@vector-consulting.de

Collaboration Tools for Global Software Engineering

Filippo Lanubile, Christof Ebert, Rafael Prikladnicki, and Aurora Vizcaíno

Software engineering involves people collaborating to develop better software. Collaboration is challenging, especially across time zones and without face-to-face meetings. We therefore use collaboration tools all along the product life cycle to let us work together, stay together, and achieve results together. Authors Filippo Lanubile, Rafael Prikladnicki, Aurora Vizcaíno, and I provide an overview of tools and technologies for improved collaboration. Our article summarizes experiences and trends chosen from recent IEEE International Conference on Global Software Engineering (IGSCE) conferences.

I look forward to hearing from both readers and prospective column authors about this column and the technologies you want to know more about. —Christof Ebert

Tools are essential to collaboration among team members, enabling the facilitation, automation, and control of the entire development process. Adequate tool support is especially needed in global software engineering because distance aggravates coordination and control problems, directly or indirectly, through its negative effects on communication.[1]

In this column, we present current collaborative development environments and tools to enable effective software development, either global or collocated.[2] Our summary is not comprehensive. Rather, we identified technologies that really matter by conducting surveys at recent ICGSE conferences (see the sidebar) and in companies where we're consulting to improve their distributed engineering capabilities.

Collaborative Development Tools

We briefly look into seven standard collaborative development tools.

Version-Control Systems

Distributed software engineering needs systematic configuration management. A version-control system lets team members share software artifacts in a controlled manner. Subversion (SVN; http://subversion.tigris.org) is a popular open source version-control system that facilitates distributed file sharing. SVN adopts a centralized architecture, in which a single central server hosts all project metadata. Developers use SVN clients to check out a limited view of the data on their local machines.

Today, several systems are using distributed version control that operates in a peer-to-peer manner. Examples include Git (www.git-scm.com), Mercurial (http://mercurial.selenic.com), and Darcs (www.darcs.net). Unlike centralized tools that let developers check out a project from a distributed version-control system, the peer-to-peer systems provide a complete clone of the project's repository (called a fork) on local machines, not a just a portion of it.

Trackers

Trackers are used to manage issues (or "tickets") such as defects, changes, or requests for support. The tracking function centers on a database that all team members can access through the Web.

Distributed trackers such as, Jira (www.atlassian.com) are a generalization of bug-tracking systems such as Bugzilla (www.bugzilla.org),

 0740-7459/10/$26.00 © 2010 IEEE

originally developed by the Mozilla project. A recorded issue includes an identifier, a description, and information about the author; it also defines a life cycle to help team members track issue resolutions.

Build Tools

The more distributed the project, the greater the need for secure, remote repository and build management. Build tools such as Maven (http://maven.apache.org) and CruiseControl (http://cruisecontrol.sourceforge.net) let projects maintain remote repositories and create and schedule workflows. The workflows facilitate continuous integration for executing scripts, compiling binaries, invoking test frameworks, deploying to production systems, and sending email notifications to developers. A Web-based dashboard shows the status of current and past builds.

Modelers

Model-based collaboration is what distinguishes collaborative software engineering from more general collaboration activities that share only files and not content.[3] Collaborative modeling tools such as Artisan Studio (www.artisansoftwaretools.com), Rational Software Modeler (www.ibm.com/software/awdtools/modeler/swmodeler), and Visible Analyst (www.visible.com/Products/Analyst) help developers create formal or semiformal software artifacts, including visual Unified Modeling Language (UML) models and customized software processes.

Knowledge Centers

These are content management systems that let team members share explicit knowledge on the Web. A knowledge center, such as the Eclipse help system (http://help.eclipse.org) or KnowledgeTree (www.ktdms.com), might contain internal documents, technical references, standards, FAQs, and best practices. Knowledge centers can also include sophisticated knowledge management activities, such as expert identification and skills management, to acquire tacit knowledge in explicit forms.

Communication Tools

Software engineers have adopted a wide range of mainstream project communication technologies when direct interaction isn't possible. Asynchronous communication tools include email, mailing lists, newsgroups, Web forums, and—more recently—blogs. Synchronous tools include standard telephone and conference calls, chat, instant messaging, voice over IP, and videoconferencing.

WebEx (www.webex.com) is the market leader for online meeting facilities. Both WebEx and WorkSpace3D (www.tixeo.com) provide a rich interface for synchronous and asynchronous collaboration. They enable voice and video over IP communication while viewing and editing documents, desktop and application sharing, co-browsing and whiteboard drawing, and meeting persistence for later replay.

The text-based eConference (http://code.google.com/p/econference) is a lean tool that supports distributed teams needing synchronous communication and structured-discussion services.[4] Such tools provide closed-group chat that's augmented with agendas, meeting minutes-editing and typing-awareness capabilities, and hand-raising panels to enable turn-based discussions.

General communication tools—that is, those that aren't specific to software engineering—fall in the category of *groupware*,[5] combining tools for document sharing and review, concurrent editing, and shared calendars. However, the term "groupware" has fallen into disuse in favor of terms such as "collaborative software" or "social software." Popular multifunction collaboration platforms are IBM Lotus Notes/Domino (www.ibm.com/software/lotus/notesanddomino) and Microsoft SharePoint (www.microsoft.com/SharePoint).

Google Wave (wave.google.com) is a recent collaboration platform designed to let people at different locations simultaneously edit documents and other artifacts without the classic checkout or delta mechanisms. However, Google Wave also reveals the limited efficiency and effectiveness of simultaneous offline collaboration because individuals have difficulty keeping track of the many parallel changes and interrupts while still trying to work on their own contributions.

Web 2.0 Applications

Web 2.0 extends traditional collaborative software by means of direct user contributions, rich interactions, and community building. Some key Web 2.0 applications are blogs, such as WordPress (http://wordpress.org); microblogs, such as Twitter (http://twitter.com); wikis, such as the Portland Pattern Repository (http://c2.com/cgi/wiki), social networking sites, such as LinkedIn (www.linkedin.com), and collaborative tagging systems, such as Delicious (http://delicious.com).

Recently, Web 2.0 applications have become quite common in open source and global software projects. They represent a valuable means to increase the informal communication exchanged among team members. For example, wiki platforms have emerged as a practical, economical option for creating and maintaining group documentation.[6]

Collaborative Development Environments

A CDE gives a project workspace with a standardized tool set for global software teams. CDEs combine several of the tools we've described and thus offer a frictionless development environment to increase developer comfort and productivity.[7] Several CDEs are available as commercial products or open source initiatives—increasingly, as online services hosted externally.

CDEs are borrowing successful features

International Conference on Global Software Engineering

Collaboration in distributed teams is pervasive in today's IT and software industry. The annual IEEE International Conference on Global Software Engineering (ICGSE) brings together researchers and practitioners interested in exploring how globally distributed teams work and how to meet the challenges they pose. Previous conferences attracted a broad audience from industry and academia. ICGSE 2010 will be held on 23–26 August 2010 in Princeton, New Jersey, USA.

For more information or to participate, see the conference Web site at www.icgse.org.

Table 1

Collaborative development environments

CDE	Collaborative development tools						
	Version-control systems	Trackers	Build tools	Modelers	Knowledge centers	Communication tools	Web 2.0 applications
SourceForge (sourceforge.net)	CVS,* SVN, Git, Mercurial, Bazaar**	Bugs, feature requests, patches, support requests	No	No	No	Mailing lists; forums	Feeds; hosted applications for blogs, microblogs and wikis
GForge (gforge.org)	CVS, SVN, Perforce†	Tasks and issues (bugs, feature requests, patches, support requests)	Integrating CruiseControl	No	Document manager	Mailing lists; forums	Feeds, wiki
Trac (trac.edgewall.org)	SVN; plug-ins for integrating Git, Perforce, Mercurial, Darcs, Bazaar	Tickets (tasks, feature requests, bugs, support issues)	Bitten plug-ins for integrating: Continuum, CruiseControl, Hudson‡	Project roadmap	As wiki	Plug-in for forums	Wiki, feeds, plug-ins for tagging tickets and wiki pages
Google Code (code.google.com)	SVN, Mercurial; integrating Git	Issues (defects, enhancements, tasks)	No	No	As wiki	Integrating Google Groups	Wiki, feeds
Assembla (www.assembla.com)	SVN, Git, Mercurial	Tickets (tasks, enhancements, ideas, defects)	No	Milestones, agile planner	As wiki	Message board, chat	Wiki, microblog, feeds
Rational Team Concert (jazz.net/projects/rational-team-concert)	Built-in	Work items (defects, enhancements, plan items, retrospectives, risk, stories, tasks, build items, use cases)	Built; integrating Ant,§ Maven	Process templates	Integrating MS SharePoint and Lotus Quickr document	Instant messaging	Feeds, wiki, tagging work items
GitHub (github.com)	Git	Issues	No	No	As wiki	No	Feeds, wiki, social networks
Launchpad (launchpad.net)	Bazaar	Bugs; integrating with external trackers	No	Blueprints (specifications of features or processes)	Questions and answers	Mailing lists	No
CodePlex (www.codeplex.com)	Built-in	Work items (features, issues, tasks)	CruiseControl.NET	Documentation tab	As wiki	Mailing lists, discussions list	Feeds, wiki

*CVS: Concurrent Versions System (www.nongnu.org/cvs)
**Bazaar (http://bazaar.canonical.com)
†Perforce (www.perforce.com)
‡Hudson (http://hudson-ci.org)
§Ant (http://ant.apache.org)

typically available on social network sites. For instance, Assembla (www.assembla.com) notifies users of project-related events via Twitter; GitHub (http://github.com) offers a Twitter-like approach to monitoring a project's progress; Rational Team Concert (www.ibm.com/software/awdtools/rtc) borrows Delicious's tagging feature, letting developers assign free keywords to managed items.

Table 1 highlights a selection of CDEs and shows how they support these tools.

Specific Collaboration Tools

CDEs are often unsuitable in companies because of legacy tools or environments that must be enhanced with specific collaboration functionalities. In these situations, developers can choose from collaboration tools that map to typical life-cycle activities.

Project Management

Collaborative project management tools

such as ActiveCollab (www.activecollab.com) and WorldView[8] offer a Web-based interface to manage project information for calendars and milestone tracking. Such tools give managers an overview of project status at different detail levels, such as team member locations and contact information. WorkSpaceActivityViewer provides an overview of ongoing project activities by using information extracted from developers' workspaces.[9]

Requirements Engineering

Major RE tools such as Doors (www.ibm.com/software/awdtools/doors) and IRqA (www.visuresolutions.com) let multiple engineers use natural language text to describe project use cases and requirements and to record dependencies among and between them. Both tools have a document-oriented, Word-based interface. They also provide a Web interface for users who need access to requirements information but not to local installations.

The collaboration tool eRequirements (www.erequirements.com) is entirely Web-based. It has abandoned a MS Word-based interface but provides Web access to collaboratively explore and manage use cases and requirements.

Design

Camel supports virtual software-design meetings by capturing and storing all design-relevant information, role definitions, and version control coordination.[10] It includes playback features to review a session once it has ended.

Prominent design and modeling tools, such as Gliffy (www.gliffy.com) and Creately (http://creately.com) support multiple diagram types such as UML or Business Process Modeling Notation. They also offer special features that simplify team communication and collaboration, such as tools for commenting, creating blogs, or even managing knowledge. Furthermore, Gliffy can be integrated with the Jira distributed tracking system.

Test

TestLink (http://testlink.sourceforge.net) is a popular tool for managing the entire testing process. It has a Web-based interface that's accessible everywhere from a browser. It organizes test cases into test plans. Users can import and execute groups of test cases by using one or more keywords that they have previously assigned to the test cases.

On the other hand, Selenium (http://seleniumhq.org) is a tool suite to automate Web application testing across many platforms. It includes an integrated development environment (IDE) for writing and running tests, a remote-control tool for controlling Web browsers on other computers, a Web-based quality-assurance tool, and an Eclipse plug-in to write Selenium and Watir (http://watir.com) tests.

Finally, OpenSTA (http://opensta.org) is a distributed software-testing architecture that can perform scripted HTTP and HTTPS heavy-load tests with performance measurements from Win32 platforms.

Trends in Collaboration Tools

New collaboration tools and associated best practices are emerging almost daily. We see two major trends. First, practically all engineering tools will provide collaboration features. These features help individual tools shared by a team, but they're implemented differently on different tools and so don't allow data integration across tools. A second, related trend is improved federation of engineering tools. Eclipse will help initially, but ensuring efficiency, consistency, and information security across multiple tools, teams, and companies finally requires a strong product life-cycle management (PLM) strategy. Tools such as Teamcenter (www.siemens.com/teamcenter) and Easee (www.vector.com/easee) allow secure federation and collaborative work with integrated data backbones.

No current tool or CDE supports all the activities necessary for global software engineering. Users must therefore prioritize their collaboration needs and the tools to support them. Introducing collaboration technology should be a stepwise process, starting with a collaboration platform to share applications. A consistent PLM strategy can evolve in parallel with this process, providing mechanisms to guide and align technologies to the degree necessary. Such a strategy is valuable when working in external networks with participants from different organizations. Within one company, users should move to a CDE as part of their overall PLM.

Effective tool support for collaboration is a strategic initiative for any company with distributed resources, no matter whether the strategy involves offshore development, outsourcing, or supplier networks. Software needs to be shared, and appropriate tool support is the only way to do this efficiently, consistently, and securely.

References

1. E. Carmel and R. Agarwal, "Tactical Approaches for Alleviating Distance in Global Software Development," *IEEE Software*, vol. 18, no. 2, 2001, pp. 22–29.
2. C. Ebert, *Global Software Engineering: Distributed Development, Outsourcing, and Supplier Management*, Wiley-IEEE CS Press, 2010.
3. J. Whitehead, "Collaboration in Software Engineering: A Roadmap," *Proc. Int'l Conf. Software Eng.* (ICSE 07), IEEE CS Press, 2007, pp. 214–225.
4. F. Calefato and F. Lanubile, "Using Frameworks to Develop a Distributed Conferencing System: An Experience Report," *Software: Practice and Experience*, vol. 39, no. 15, 2009, pp. 1293–1311.
5. C.A. Ellis, S.J. Gibbs, and G. Rein, "Groupware: Some Issues and Experiences," *Comm. ACM*, vol. 34, no. 1, 1991, pp. 39–58.
6. P. Louridas, "Using Wikis in Software Development," *IEEE Software*, vol. 23, no. 2, 2006, pp. 88–91.
7. G. Booch and A.W. Brown, "Collaborative Development Environments," *Advances in Computers*, vol. 59, 2003, pp. 2–29.
8. A. Sarma and A. van der Hoek, "Towards Awareness in the Large," *Proc. Int'l Conf. Global Software Engineering* (ICGSE 06), IEEE CS Press, 2006, pp. 127–131.
9. R. Ripley, A. Sarma, and A. van der Hoek, "A Visualization for Software Project Awareness and Evolution," *Proc. Int'l Workshop on Visualizing Software for Understanding and Analysis* (VISSOFT 2007), pp. 137–144.
10. M. Cataldo et al., "CAMEL: A Tool for Collaborative Distributed Software Design," *Proc. Int'l Conf. Global Software Eng.* (ICGSE 09), IEEE CS Press, 2009, pp. 83–92.

Filippo Lanubile is an associate professor of computer science at the University of Bari. Contact him at lanubile@di.uniba.it.

Christof Ebert is managing director and partner at Vector Consulting Services, a consulting and research firm focused on improving technical product development. Contact him at christof.ebert@vector.com.

Rafael Prikladnicki is an assistant professor at the Computer Science School and project manager coordinator at the Technological Management Agency at Pontifícia Universidade do Rio Grande do Sul (PUCRS). Contact him at rafaelp@pucrs.br.

Aurora Vizcaino is an associate professor in the University of Castilla-La Mancha's Computer Science Department. Contact her at aurora.vizcaino@uclm.es.