Building the Community of Leading Software Practitioners

Technologies to Support Collaboration across Time Zones

Rafael Prikladnicki, Sabrina Marczak, Erran Carmel, and Christof Ebert

from the special issue on Software Engineering for Compliance
Vol. 29, No. 3
May/June 2012

www.computer.org/software
Technologies to Support Collaboration across Time Zones

Rafael Prikladnicki, Sabrina Marczak, Erran Carmel, and Christof Ebert

Time zone differences are a challenge to global software engineering. This column surveys the key technologies and tools that support collaboration across time zones. The insights on technologies derive from a meta-analysis of the 2010 and 2011 IEEE International Conference on Global Software Engineering (ICGSE), among others. I look forward to hearing from both readers and prospective authors about this column and the technologies and tools you want to know more about. — Christof Ebert

Collaboration across time zones depends on technology for communicating, coordinating, and staying aware of what's going on with the project. But even the fanciest videoconferencing or 3G holography can't overcome the fundamental time-zone problem—that it's sleep time on the other side of the world. Consequently, no miracle technology will overcome time-zone differences. However, better understanding and use of the available technologies will help mitigate the challenges that time-zone separation incurs. This column presents technologies that support collaboration among globally distributed software teams. It also goes beyond the dichotomy of synchronous versus asynchronous by classifying technology use in a new work mode: scattertime. Scattertime emerged with the Internet and mobility, where knowledge workers spread their labor into scattered chunks of time, broken up by home and personal time. For instance, a developer on a far-flung team might not share any overlapping hours with her colleagues, so she time-shifts her work schedule to call in to weekly group meetings late at night from home. Her flexibility lets her synchronously meet with the team. From this new perspective, collaboration can take place anywhere and at any time.

Better understanding and use of the available technologies will help mitigate the challenges.

Supporting Time-Zone Collaboration

Papers from the 2010 and 2011 IEEE International Conferences on Global Software Engineering (ICGSE) identified technologies that can support
## Technologies and their uses.

<table>
<thead>
<tr>
<th>Technology</th>
<th>Collaboration type</th>
<th>Typical use in software engineering community</th>
<th>Emerging use by distributed software development teams</th>
<th>Tools</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Software engineering and project management tools</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Project management tools</td>
<td>Synchronous (Sync), scattertime (Scat), and asynchronous (Async)</td>
<td>Share and assign task assignments and status, progress, schedule, calendar and members’ agendas</td>
<td>Management of the project is pushed into the hands of the entire team</td>
<td>ActiveCollab, Basecamp, Microsoft Project, and WorldView</td>
</tr>
<tr>
<td>Software version control systems</td>
<td>Sync, Scat, and Async</td>
<td>Concurrent work on the same source code in a controlled way; allows members to revert code version after a discussion</td>
<td>Distributed version control systems that clone the entire project repository on the developer’s machine (instead of portions, as in traditional centralized systems)</td>
<td>Bazaar, CVS, git, and SVN</td>
</tr>
<tr>
<td>Issue- and defect-tracking systems</td>
<td>Sync, Scat, and Async</td>
<td>Maintaining the list of issues, defects, changes, requests, and problems</td>
<td>Assigning people and dates for resolution of logged issues and defects</td>
<td>Bugzilla, Jira, and Trac</td>
</tr>
<tr>
<td>Build systems</td>
<td>Sync, Scat, and Async</td>
<td>Automate software release and deployment.</td>
<td>Deliver fully tested code as quickly as possible and in small iterations.</td>
<td>Apache Ant and CMake</td>
</tr>
<tr>
<td>Continuous integration systems</td>
<td>Sync, Scat, and Async</td>
<td>Daily integration of code in single-source repository</td>
<td>Hourly integration to keep everyone in sync</td>
<td>CruiseControl, Hudson, Rational Team Concert, and Team Foundation Server</td>
</tr>
<tr>
<td><strong>Web 2.0 tools</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Virtual whiteboards</td>
<td>Scat</td>
<td>Conduct creativity-based activities</td>
<td>Serve as task boards for distributed agile teams aiding project management</td>
<td>Dabbleboard, Scriblink, Scribblar, and Skrbl</td>
</tr>
<tr>
<td>Microblog</td>
<td>Sync, Scat, and Async</td>
<td>Task status, announcement of project events, and problem resolution</td>
<td>Integrated into software development environments for intra- and interproject knowledge acquisition and learning</td>
<td>In-house tools and Twitter</td>
</tr>
<tr>
<td>Blog</td>
<td>Scat and Async</td>
<td>Post news and collaborate; share experiences</td>
<td>Share concepts and record team decisions; shared work diaries</td>
<td>Blogger and Wordpress</td>
</tr>
<tr>
<td>Wiki</td>
<td>Scat and Async</td>
<td>Concentrate quick contribution of colleagues; knowledge sharing</td>
<td>Allows numerous stakeholders to get involved in requirements engineering process</td>
<td>Confluence and Mediawiki</td>
</tr>
<tr>
<td><strong>General-purpose tools</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Telepresence</td>
<td>Sync</td>
<td>Formal meetings in large firms</td>
<td>Simulate continuous collocation in real time</td>
<td>Cisco Telepresence Magic</td>
</tr>
<tr>
<td>Screen sharing</td>
<td>Sync and Scat</td>
<td>For meetings and discussions that involve documentation, graphical representations, and demonstrations</td>
<td>Distributed agile teams: paired programming and code reviews</td>
<td>Mikogo and Skype Screen Sharing</td>
</tr>
<tr>
<td>Videoconference</td>
<td>Sync and Scat</td>
<td>For group discussions, presentations, and document reviews</td>
<td>Team kick-off puts a face to each person; always-on video simulates proximity</td>
<td>Cisco WebEx, iVisit, Megameeting, and PalTalk</td>
</tr>
<tr>
<td>Voice over IP and voice conference</td>
<td>Sync and Scat</td>
<td>Save money on landline and mobile calls</td>
<td>Time shifting to allow synchronous voice communication</td>
<td>Google phone feature, over the phone, and Skype</td>
</tr>
<tr>
<td>Instant messaging</td>
<td>Sync and Scat</td>
<td>For informal situations such as availability notification, quick expert queries, brainstorming, and sharing unfinished ideas.</td>
<td>For convenient individual voice and video conversations</td>
<td>Cisco Unified Presence, IBM Lotus, Jabber XCP, Microsoft Live Messenger, and Sametime</td>
</tr>
<tr>
<td>Email</td>
<td>Sync, Scat, and Async</td>
<td>Everything at any time</td>
<td>Email integrated with awareness tools (for example, to view people available to chat and find the best available time to schedule meetings across time zones)</td>
<td>Gmail, Microsoft Exchange, and Thunderbird</td>
</tr>
</tbody>
</table>
collaboration across time zones (see the ICGSE sidebar). In *Global Software and IT: A Guide to Distributed Development, Projects, and Outsourcing*, Christof Ebert provided a basic framework for such collaboration technologies.2 For this article, we also examined data on Web 2.0 tools in the literature by reviewing studies published at the Workshop on Web 2.0 for Software Engineering (WebSE) in conjunction with the International Conference on Software Engineering (ICSE; see https://sites.google.com/site/web2se2011 and https://sites.google.com/site/web2se), and interviewed globally distributed team members of large multinational IT companies.3 Most of the technologies we identified are familiar, so we’ll focus on their usage among global software engineers and opportunities for deeper use.

Table 1 presents the landscape of technologies along with their current and emerging usage and some examples of tools.

**Reported Technology Adoption**

Despite the advances of Web 2.0 technologies, reports list email and instant messaging as the most effective and adopted communication tools.2,3 Distributed team members find it useful to adopt project-wide mailing lists to share important issues with the entire team. Another benefit of using mailing lists is that team members can easily retrieve information from their mailboxes. Instant messaging is a good resource when you need a quick reply to something and you know your colleague is available. Projects use audio conferencing for weekly team meetings to coordinate work (for example, to report progress, discuss challenges, present new ideas, and share awareness). Videoconferencing is less useful because it demands some tuning and availability can be reduced in areas with limited bandwidth.

Surprisingly, the use of Web 2.0 technology isn’t that widespread among global software engineering teams. Wikis play some role for collaborative requirements elicitation among distributed stakeholders but often result in a growing, incomprehensive heap of information. Modeling tools still lack features to support coordination and synchronization of artifacts among several distributed team members working together in the same modeling-task-related activity.

A variety of tools support collaboration among distributed team members. However, some organizations still claim that these tools don’t deliver sufficient value. The reasons often stem from a lack of tools strategy in the companies, where tools just organically evolve. Thus, the tools don’t support teams and don’t address the three major concerns of distributed teams—namely, performance, integrity, and security. Previous generations of collaboration tools were focused on asynchronous communication, traditional client-server applications, and proprietary systems. The current generation of collaboration tools have integrated voice, video, and visual communication, along with a combination of Web-based and traditional client-server architecture that allow synchronous and asynchronous communication to support scattertime.

Mere application life-cycle management and product life-cycle management certainly aren’t the answer, but they will help in finding an overall information architecture. The federation of engineering tools will continue to grow. Eclipse will help initially, but ensuring performance, integrity, and information security across multiple tools, teams, and companies requires a sustainable collaboration strategy. Figure 1 provides an example for combining requirements engineering, project management, programming, and communication tools together in a single environment, letting developers collaborate by using integrated planning, source control, work items, build, dashboards, reports, and process support. Rational Team Concert (RTC) is an example of a collaborative application life-cycle management where software development teams can manage all aspects of their work, such as plans, tasks, source control, build management, and reports. RTC provides a single, integrated environment for several aspects of the software development process, where users can share team information, post detailed information, and configure which information is visible.
When integration isn’t possible, using a set of tools might be the alternative. For example, the combined use of a whiteboard with a video camera can support a collaborative software design session by live streaming what’s being drawn on the board and using a shared document (such as Google Docs) to collaboratively take notes during the meeting. Web 2.0 technologies have facilitated such synchronous collaboration. In addition, mobile and cloud tools enable scatternwork—people can work when commuting, watching TV, from home, and so on.

No current tool 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 life-cycle management strategy that connects processes, tools, and information-sharing needs is indispensable when working in external networks with participants from different organizations.

References

RAFAEL PRIKLADNICKI is an associate professor in the Computer Science School and director of the Technology Management Agency (AGT) at Pontifícia Universidade do Rio Grande do Sul (PUCRS) in Porto Alegre, Brazil. Contact him at rafaelp@pucrs.br or visit his website at www.inf.pucrs.br/~rafael.

SABRINA MARCZAK is an associate professor in the Computer Science School at Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS) in Porto Alegre, Brazil. Contact her sabrina.marczak@pucrs.br.

ERRAN CARMEL is a professor in the Information Technology Department, Kogod School of Business at the American University in Washington, DC, where he is also an international business research professor. Contact him at carmel@american.edu.

CHRISTOF EBERT is managing director at Vector Consulting Services. He is a senior member of IEEE and the editor of this department for IEEE Software magazine. Contact him at christof.ebert@vector.com.
Two Special ICGSE 2012 offers!

A professional peer-reviewed magazine featuring
- Broad coverage of technologies, processes, and management issues
- Practical advice
- Experiences with the latest innovations
- Clear take-away messages

Recent and upcoming themes include
- The Software Business
- Cloud Computing
- Multicore Programming for the Desktop
- Software Protection
- Algorithms for Today’s Practitioner
- Software Engineering for Climate Change

Check out our special offers—now available in print and digital!

Check out an entire free issue: www.qmags.com/ISW/free-icgse12

To subscribe to 6 issues of the digital edition:
www.qmags.com/ISW/ICGSE12

US $19.95/year