Global Software and IT

A Guide to Distributed Development, Sourcing, and Outsourcing

Christof Ebert

2010

1 Case Study: Collaborative Development Environments

Authors: Fabio Calefato and Filippo Lanubile, University of Bari

Motivation

This chapter provides a case study from different companies and shows how to best use tools in globally distributed software projects. The case study highlights relevant themes and guidance from previous chapters in a concrete project context. It offers valuable insights towards how to do things in your own company.

Adequate tool support is paramount to enable collaboration between team members and to control the overall development process. This is especially true in global software engineering because of distance [Herbsleb01]. Distance has an impact on the three main forms of cooperation within a team [Carmel01]: communication coordination, and control. Communication is the exchange between the members of information, whether formal or informal, occurring in planned or impromptu interaction. Coordination is that act of orchestrating each task and organizational unit, so that they all contribute to the overall objective. Control is the process of adhering to goals, policies, standards or quality levels, set either formally (e.g., formal meetings, plans, guidelines) or informally (e.g., team culture, peer pressure). Distributed teams create further burdens on communication, coordination and control mechanisms, primarily the informal ones.

Due to distance, people cannot coordinate and control by just visiting the other team members. The absence of management-by-walking can result in coordination and control issues, like misalignment and rework. When control and coordination needs of distributed software teams rise, so does the load on all communication channels available. In fact, software projects have two complementary communication needs. First, the more formal, official communications is used for crucial tasks like updating project status, escalating project issues, and determining who has responsibility for particular work products. Secondly, informal ‘corridor talk’ allows team members to keep a ‘peripheral awareness’ of what is going on around them, what other people are working on, what states the various parts of the project are in, and many other essential pieces of background information that enable developers to work together efficiently. In colocated settings, communication is taken for granted and then, its importance often goes unnoticed. When developers are not located together, they have much less opportunities of communication. There is empirical evidence that the frequency of communication drops off with the physical separation among developers’ sites [Herbsleb03]. Therefore, distance exacerbates coordination and control problems directly or indirectly through its negative effects on communication. In other words, communication disruption due to distance further increases and aggravates coordination and control breakdowns [Carmel01].

Distance can have an effect on three distinct dimensions: geographical, temporal, and socio-cultural. Geographical distance is a measure of the spatial dispersion, occurring when team members are scattered across different sites. It can be operationalized as the cost or effort required to exchange visits from one site to another. Temporal distance is a measure of the temporal dispersion, occurring when team members wishing to interact. It can be caused by time-zone differences or just time shifting work patterns (e.g., one site having a quick lunch break at noon and another site a two-hour lunch time at 1:00 pm). Socio-cultural distance is a measure of the effort required by team members to understand the organizational and national cultures (e.g., norms, practices, values, spoken languages) in remote sites.

Cooperation difficulties due to distance can only be partially tackled using appropriate techniques. For instance, coordination and control issues can be counteracted, respectively, adopting architectural frameworks that enable a better division of labor between teams, and choosing an agile development process. However, global development would not just be feasible without adequate tool support [Ebert06]. In fact, developers need constant tool support during the whole software life-cycle, in order to model, design, and test software functionalities, manage a myriad of interdependent artifacts, and communicate with each other. In the next section we present a number of tools and collaborative development environments that are available today to enable effective global software development.

Background

Tools provide a considerable help to software development activities. Software engineering tools to assist distributed projects may fall into the following categories: software configuration management, bug and change tracking, build and release management, modelers, knowledge centers, communication tools, and collaborative development environments.

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 to share software artifacts with other team members in a controlled manner. Rather than just using a directory to exchange files with other people, developers can use an SCM tool to be sure that interdependent files are changed together and control who is allowed to make changes. Further, SCM tools make it possible to save messages about what changed and why. Open-source SCM tools have become indispensable tools for coordinating the interaction of distributed developers. Until early 2000s, the world of SCM tools has been quite stale [O'Sullivan09]. Released in 1990, Concurrent Version System (CVS)[1] is the ancestor of the many open source SCM tools available today and, despite of some severe drawbacks (e.g., limitations in renaming and deleting folders), it is still in wide use today although as a legacy system. Subversion (SVN)[2] came out a decade after CVS with the goal of overcoming the negative aspects of CVS. Both SVN and CVS adopt a centralized, client-server approach. A single central server hosts all project's metadata. Developers check out from the central server a limited view of the data on their local machines. In early 2000s, however, a number of projects (e.g., Git[3], Mercurial[4], and Darcs[5]) were started to develop distributed SCM tools that operate in a peer to peer manner..

[1] http://www.nongnu.org/cvs/

[2] http://subversion.tigris.org/

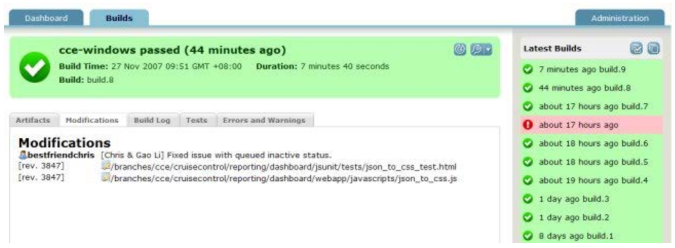
[3] http://git-scm.com/

[4] http://mercurial.selenic.com/wiki/

[5] http://darcs.net/

Bug and change tracking is centered around 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. Bug tracking systems, such as Bugzilla[6] and JIRA[7], also define a life-cycle for bugs to help team members to track the resolution of defects. 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 allows projects to create and schedule 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 function. Build and release management tools can also provide a web-based dashboard to view the status of current and past builds (Fig. 1). Build tools, such as CruiseControl[8] and its ancestor like the UNIX make utility, are essential tools to perform Continuous Integration [Fowler06], an agile development practice which allows developers to integrate daily thus reducing integration problems.



cce-windows passed (44 minutes ago)
Build Time: 27 Nov 2007 09:51 GMT +08:00 Duration: 7 minutes 40 seconds
Build: build.8

Modifications
bestfriendchris [Chris & Gao Li] Fixed issue with queued inactive status.
[rev. 3847] /branches/cce/cruisecontrol/reporting/dashboard/jsunit/tests/json_to_css_test.html
[rev. 3847] /branches/cce/cruisecontrol/reporting/dashboard/webapp/javascripts/json_to_css.js

Latest Builds
- 7 minutes ago build.9
- 44 minutes ago build.8
- about 17 hours ago build.7
- about 17 hours ago
- about 18 hours ago build.6
- about 18 hours ago build.5
- about 19 hours ago build.4
- 1 day ago build.3
- 1 day ago build.2
- 8 days ago build.1

Fig. 1: Project build information within a dashboard

Model-based collaboration is what distinguishes collaborative software engineering from more general collaboration activities which only share files and not content [Whitehead07]. Collaborative modeling tools such as Artisan Studio[9], Rational Software Modeler[10] and Visible Analyst[11] help developers to create formal or semiformal software artifacts including visual UML modeling software artifacts and customized software processes.

Product and process modeling encompasses the core features of what was called Computer Aided Software Engineering (CASE), from requirements engineering to visual modeling of both software artifacts and customized software processes. Collaboration in software development tends to be around the creation of formal or semiformal software artifacts. According to [Whitehead07], model-based collaboration is what distinguishes software engineering collaboration from more general collaboration activities which lack the focus on using the models to create shared meanings.

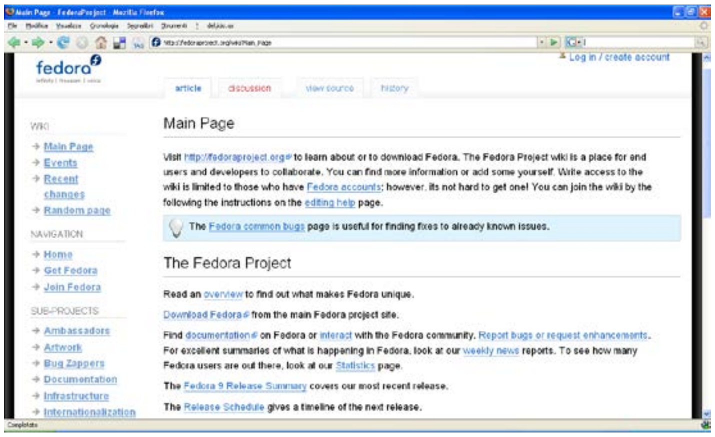
Knowledge centers are mostly document-driven and web-enabled, and allows team members to share explicit knowledge across a work unit. A knowledge center includes technical references, standards, frequently asked questions (FAQs) and best practices. Using wiki

[6] http://www.bugzilla.org/
[7] http://www.atlassian.com/software/jira/
[8] http://cruisecontrol.sourceforge.net/
[9] http://www.artisansoftwaretools.com/products/
[10] http://www-01.ibm.com/software/awdtools/modeler/swmodeler/
[11] http://www.visible.com/Products/Analyst/

software for collaborative web publishing has emerged as a practical and economical option to consider for creating and maintaining group documentation. Wikis are particularly valuable in distributed projects as global teams may use them to organize, track, and publish their work [Louridas06]. Fig. 2 shows the home page of the Fedora project wiki where both developers and users may contribute other than find information. Knowledge centers may also include sophisticated knowledge management activities to acquire tacit knowledge in explicit forms, such as expert identification and skills management [Rus02].

Communication tools increase productivity in global teams. Software engineers have adopted a wide range of mainstream communication technologies for project use in addition or replacement of communicating face-to-face by speech. Asynchronous communication tools include email, mailing lists, newsgroups, web forums and blogs; synchronous tools include the classic telephone and conference calls, chat, instant messaging, voice over IP, and videoconferencing. 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' [Erickson00] in that it does not enable users to signal their availability. Before becoming an indispensable tool ubiquitous in every workplace, email was initially used by the niche of research community and opposed by management. Likewise, chat and instant messaging have followed a similar evolution path. At first mostly used by young people for exchanging 'social' messages, these synchronous tools have spread more and more in the workplace. While email is socially blind, these tools, in contrast, provide a lightweight means to ascertain availability of remote team members and contact them in a timely manner.

Communication in distributed development can be supported by providing stakeholders with a variety of different options. Do not expect one tool to fit all. Many sites involved mean many different culture, habits, most of all, language skills.



Main Page

Visit http://fedoraproject.org to learn about or to download Fedora. The Fedora Project wiki is a place for end users and developers to collaborate. You can find more information or add some yourself. Write access to the wiki is limited to those who have Fedora accounts; however, its not hard to get one! You can join the wiki by the following the instructions on the editing help page.

The Fedora common bugs page is useful for finding fixes to already known issues.

The Fedora Project

Read an overview to find out what makes Fedora unique.

Download Fedora from the main Fedora project site.

Find documentation on Fedora or interact with the Fedora community. Report bugs or request enhancements. For excellent summaries of what is happening in Fedora, look at our weekly news reports. To see how many Fedora users are out there, look at our Statistics page.

The Fedora 9 Release Summary covers our most recent release.

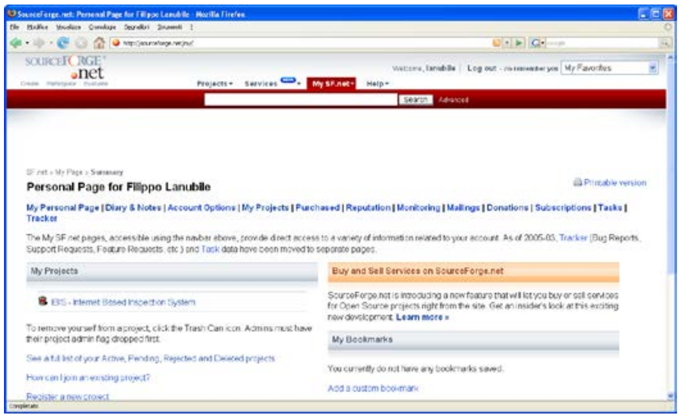
The Release Schedule gives a timeline of the next release.

Fig. 2: Fedora Project documentation based on wiki

General communication tools (i.e., non software engineering specific) fall in the category of ‘groupware’ which refers to the class of applications that support groups of people engaged in performing a common task [Ellis91]. However, the term groupware is nowadays almost disused in favor of preferred wordings such as ‘collaborative software’, ‘social software,’ or ‘Web 2.0’ [Murugesan07], which also include systems used outside the workplace (e.g., blogs, wikis, instant messaging).

Interoperability and a familiar user interface provide strong motivations to integrate task-specific solutions and generic groupware into collaborative development environments (CDE). 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 CDE are available as commercial products, open source initiatives or prototypes to enable distributed software development.

SourceForge[12]is the most popular CDE with over 230.000 hosted projects and 2 million registered users, as of this writing. The original mission of SourceForge was to enrich the open source community by providing a centralized place for developers to control and manage OSS projects. SourceForge offers a variety of free services: web interface for project administration, space for web content and scripts, trackers (for reporting bugs, submitting support requests or patches to review, and posting feature requests), mailing lists and discussion forums, download notification of new releases, shell functions and compile farm, and supports CVS, Subversion, Git, Mercurial, and Bazaar[13] configuration management tools. Fig. 3 shows the personal page of the author which provides access to a standard toolset which can be used on every project. The commercial versions for corporate use, called SourceForge Enterprise Edition and CollabNet Enterprise Edition, add features for tracking, measuring and reporting on software project activities.


SourceForge.net: Personal Page for Filippo Lanubile - Mozilla Firefox
SOURCEFORGE .net
Welcome, lanubile | Log out
My Favorites
Projects
Services
My SF.net
Help
Search
Advanced
SF.net > My Page > Summary
Printable version

Personal Page for Filippo Lanubile

My Personal Page | Diary & Notes | Account Options | My Projects | Purchased | Reputation | Monitoring | Mailings | Donations | Subscriptions | Tasks | Tracker

The My SF.net pages, accessible using the navbar above, provide direct access to a variety of information related to your account. As of 2005-03, Tracker (Bug Reports, Support Requests, Feature Requests, etc.) and Task data have been moved to separate pages.

My Projects

IBIS - Internet Based Inspection System

To remove yourself from a project, click the Trash Can icon. Admins must have their project admin flag dropped first.

See a full list of your Active, Pending, Rejected and Deleted projects

How can I join an existing project?

Register a new project

Buy and Sell Services on SourceForge.net

SourceForge.net is introducing a new feature that will let you buy or sell services for Open Source projects right from the site. Get an insider's look at this exciting new development. Learn more »

My Bookmarks

You currently do not have any bookmarks saved.

Add a custom bookmark

Fig. 3: Personal SourceForge portal

GForge[14] is a fork of the SourceForge.net project. It has been downloaded and configured as in-house server by many industrial and academic organizations (see Fig. 4). Like SourceForge it also offers a commercial version, called GForge Advanced Server. It supports

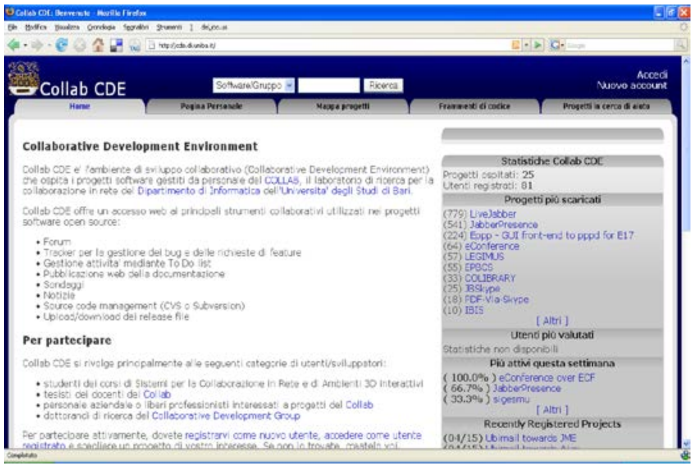
[12] http://sourceforge.net/

[13] http://bazaar.canonical.com/

[14] http://gforge.org/projects/gforge/

CVS, Subversion, and Perforce[15] configuration management tools. A notable feature of GForge is the integration with the CruiseControl build tool.

Ohloh[16] is an online community platform built upon a web services suite. Its aim is to map the status of OSS development world by retrieving data from public CDEs (Fig. 5). As such, Ohloh provides statistics about projects longevity, licenses, and software measurements, such as source lines of code and commit statistics, so as to inform about the amount of activity for each project. It also allows evaluating trend popularity of specific programming languages through global statistics per language measures. Contributor statistics are also available, meant to measure developers' own experience on the basis of commit statistics and mutual ratings (in form of "kudos" received from other developers in the community). As of January 2010, Ohloh counts over 440,000 members and lists over 430,000 projects.



Collaborative Development Environment

Collab CDE e' l'ambiente di sviluppo collaborativo (Collaborative Development Environment) che ospita i progetti software gestiti da personale del COLLAB, il laboratorio di ricerca per la collaborazione in rete del Dipartimento di Informatica dell'Universita' degli Studi di Bari.

Collab CDE offre un accesso web ai principali strumenti collaborativi utilizzati nei progetti software open source:

- Forum
- Tracker per la gestione del bug e delle richieste di feature
- Gestione attivita' mediante To Do list
- Pubblicazione web della documentazione
- Sondaggi
- Notizie
- Source code management (CVS o Subversion)
- Upload/download dei release file

Per partecipare

Collab CDE si rivolge principalmente alle seguenti categorie di utenti/sviluppatori:

- studenti dei corsi di Sistemi per la Collaborazione in Rete e di Ambienti 3D Interattivi
- tesisti dei docenti del Collab
- personale aziendale o liberi professionisti interessati a progetti del Collab
- dottorandi di ricerca del Collaborative Development Group

Per partecipare attivamente, dovete registrarvi come nuovo utente, accedere come utente registrato e sceglierе un progetto di vostro interesse. Se non lo trovate, createlo voi.

Statistiche Collab CDE

Progetti ospitati: 25
Utenti registrati: 81

Progetti più scaricati

(779) LiveJabber
(541) JabberPresence
(224) Eppp - GUI front-end to pppd for E17
(64) eConference
(57) LEGIMUS
(55) EPBCS
(33) COLIBRARY
(25) JBSkype
(18) FDF-Via-Skype
(10) IBIS
[Altri]

Utenti più valutati

Statistiche non disponibili

Più attivi questa settimana

(100.0%) eConference over ECF
(66.7%) JabberPresence
(33.3%) sigesmu
[Altri]

Recently Registered Projects

(04/15) Ubimail towards JME

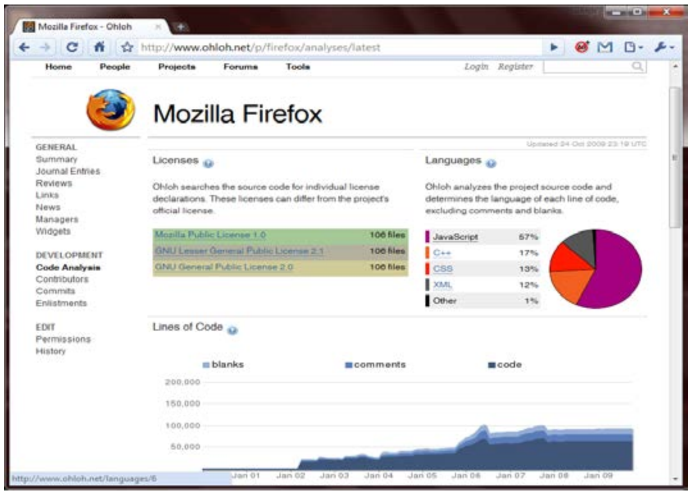
Fig. 4: A GForge-based CDE

Trac[17] is a CDE that combines an integrated wiki, an issue tracking system and a front-end interface to SCM tools, usually Subversion, although it supports a number of other configuration management tools through plug-ins. Also CruiseControl can be integrated via plug-ins to support source code building. 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. Fig. 6 shows a screenshot of a project with active tickets grouped by milestone and colored to indicate different priorities.

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

[16] http://www.ohloh.net/

[17] http://trac.edgewall.org/



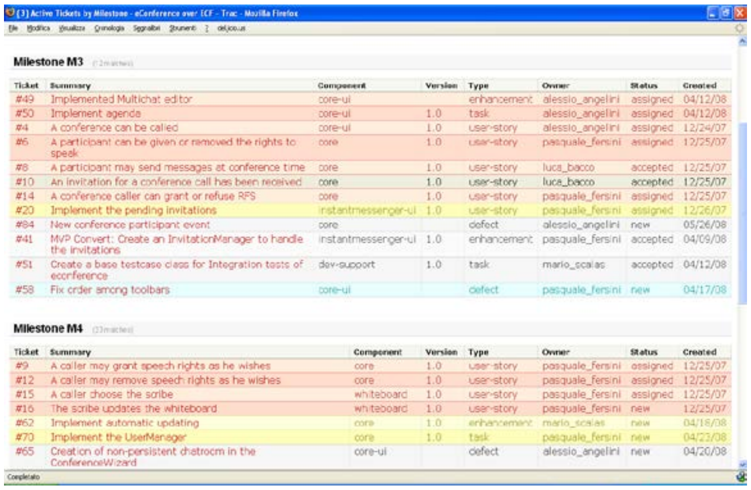
Ohloh searches the source code for individual license declarations. These licenses can differ from the project's official license.

Mozilla Public License 1.0	106 files
GNU Lesser General Public License 2.1	106 files
GNU General Public License 2.0	106 files

Ohloh analyzes the project source code and determines the language of each line of code, excluding comments and blanks.

JavaScript	57%
C++	17%
CSS	13%
XML	12%
Other	1%

Fig. 5: Ohloh's statistics non Mozilla Firefox code base


(3) Active Tickets by Milestone - eConference over ECF - Trac - Mozilla Firefox
File
Modifica
Visualizza
Cronologia
Segnalibri
Strumenti
?
del.icio.us
Completato

Milestone M3 (12 matches)

Ticket	Summary	Component	Version	Type	Owner	Status	Created
#49	Implemented Multichat editor	core-ui		enhancement	alessio_angelini	assigned	04/12/08
#50	Implement agenda	core-ui	1.0	task	alessio_angelini	assigned	04/12/08
#4	A conference can be called	core-ui	1.0	user-story	alessio_angelini	assigned	12/24/07
#6	A participant can be given or removed the rights to speak	core	1.0	user-story	pasquale_fersini	assigned	12/25/07
#8	A participant may send messages at conference time	core	1.0	user-story	luca_bacco	accepted	12/25/07
#10	An invitation for a conference call has been received	core	1.0	user-story	luca_bacco	accepted	12/25/07
#14	A conference caller can grant or refuse RFS	core	1.0	user-story	pasquale_fersini	assigned	12/25/07
#20	Implement the pending invitations	instantmessenger-ui	1.0	user-story	pasquale_fersini	assigned	12/26/07
#84	New conference participant event	core		defect	alessio_angelini	new	05/26/08
#41	MVP Convert: Create an InvitationManager to handle the invitations	instantmessenger-ui	1.0	enhancement	pasquale_fersini	accepted	04/09/08
#51	Create a base testcase class for Integration tests of econference	dev-support	1.0	task	mario_scalas	accepted	04/12/08
#58	Fix order among toolbars	core-ui		defect	pasquale_fersini	new	04/17/08

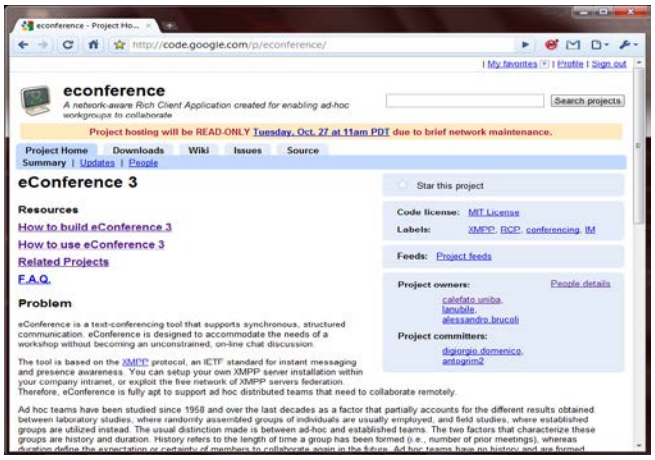
Milestone M4 (13 matches)

Ticket	Summary	Component	Version	Type	Owner	Status	Created
#9	A caller may grant speech rights as he wishes	core	1.0	user-story	pasquale_fersini	assigned	12/25/07
#12	A caller may remove speech rights as he wishes	core	1.0	user-story	pasquale_fersini	assigned	12/25/07
#15	A caller choose the scribe	whiteboard	1.0	user-story	pasquale_fersini	assigned	12/25/07
#16	The scribe updates the whiteboard	whiteboard	1.0	user-story	pasquale_fersini	new	12/25/07
#62	Implement automatic updating	core	1.0	enhancement	mario_scalas	new	04/16/08
#70	Implement the UserManager	core	1.0	task	pasquale_fersini	new	04/23/08
#65	Creation of non-persistent chatroom in the ConferenceWizard	core-ui		defect	alessio_angelini	new	04/20/08

Fig. 6: Active tickets in Trac grouped by milestone

Google Code[18] is a Google application that offers a project hosting service with revision control (only SVN and Mercurial are supported), issue tracking, a wiki for documentation, and a file download features (Fig. 7). Google code service is free for all OSS projects that are licensed under one of the following nine licenses: Apache, Artistic, BSD, GPLv2, GPLv3, LGPL, MIT, MPL, and EPL. The site also limits the maximum number of projects that a single developer can create.

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


econference - Project Ho...
http://code.google.com/p/econference/
| My favorites | Profile | Sign out
Search projects
Project Home
Downloads
Wiki
Issues
Source
Summary | Updates | People
Star this project

econference

A network-aware Rich Client Application created for enabling ad-hoc workgroups to collaborate

Project hosting will be READ-ONLY Tuesday, Oct. 27 at 11am PDT due to brief network maintenance.

eConference 3

Resources

How to build eConference 3

How to use eConference 3

Related Projects

F.A.Q.

Problem

eConference is a text-conferencing tool that supports synchronous, structured communication. eConference is designed to accommodate the needs of a workshop without becoming an unconstrained, on-line chat discussion.

The tool is based on the XMPP protocol, an IETF standard for instant messaging and presence awareness. You can setup your own XMPP server installation within your company intranet, or exploit the free network of XMPP servers federation. Therefore, eConference is fully apt to support ad hoc distributed teams that need to collaborate remotely.

Ad hoc teams have been studied since 1958 and over the last decades as a factor that partially accounts for the different results obtained between laboratory studies, where randomly assembled groups of individuals are usually employed, and field studies, where established groups are utilized instead. The usual distinction made is between ad-hoc and established teams. The two factors that characterize these groups are history and duration. History refers to the length of time a group has been formed (i.e., number of prior meetings), whereas

Code license: MIT License

Labels: XMPP, RCP, conferencing, IM

Feeds: Project feeds

Project owners: People details

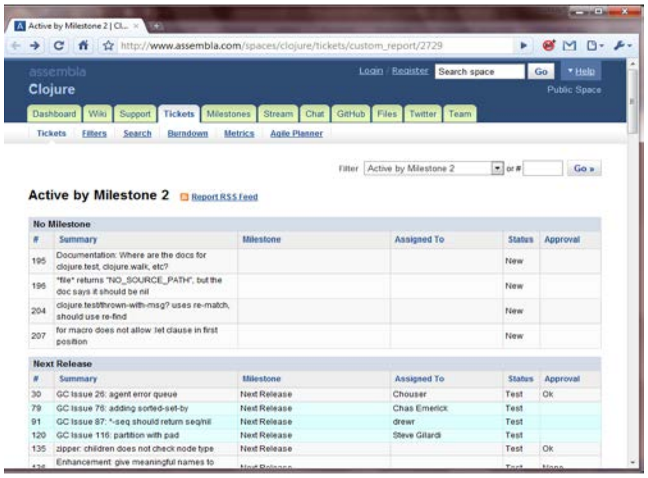
calefato.uniba,
lanubile,
alessandro.brucoli

Project committers:

digiorgio.domenico,
antognm2

Fig. 7: An example of project summary page in Google Code

Assembla[19] is yet another CDE service for both open source and commercial software (Fig. 8). Other than offering the most common features of a typical CDE, Assembla distinguishes itself from other environments for a few noticeable aspects, namely the chance to choose between SVN, Git, and Mercurial for software configuration management, the notification of changes also available via Twitter, and the support offered to teams adopting an agile development process for running Scrum meetings [Schwaber01].



Clojure

Filter: Active by Milestone 2

Active by Milestone 2

No Milestone

#	Summary	Milestone	Assigned To	Status	Approval
195	Documentation: Where are the docs for clojure.test, clojure.walk, etc?			New	
196	*file* returns "NO_SOURCE_PATH", but the doc says it should be nil			New	
204	clojure.test/thrown-with-msg? uses re-match, should use re-find			New	
207	for macro does not allow :let clause in first position			New	

Next Release

#	Summary	Milestone	Assigned To	Status	Approval
30	GC Issue 26: agent error queue	Next Release	Chouser	Test	Ok
79	GC Issue 76: adding sorted-set-by	Next Release	Chas Emerick	Test	
91	GC Issue 87: *-seq should return seq/nil	Next Release	drewr	Test	
120	GC Issue 116: partition with pad	Next Release	Steve Gilardi	Test	
135	zipper: children does not check node type	Next Release		Test	Ok
[illegible]	Enhancement: give meaningful names to	Next Release		Test	None

Fig. 8: Active tickets in Assembla grouped by milestone

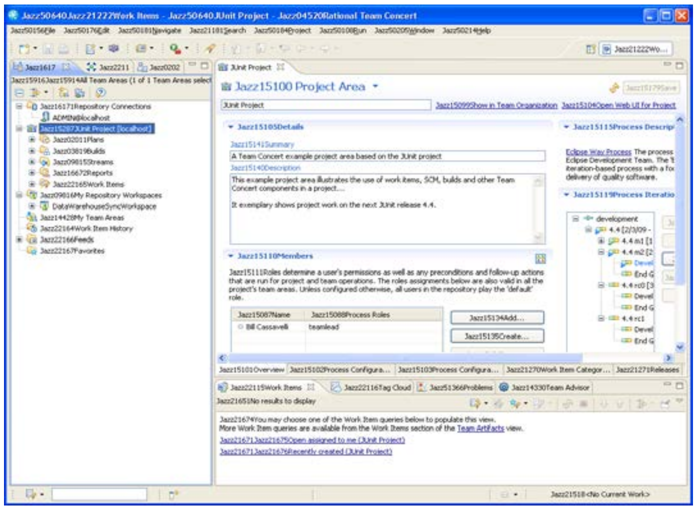
Jazz [Frost07] is an extensible platform which leverages the Eclipse notion of plug-ins to build specific CDE products like the IBM Rational Team Concert[20] (Fig. 9). The present version has a wide-ranging scope but in the former version of Jazz [Cheng04, Hupfer04] the goal was to integrate synchronous communication and reciprocal awareness of coding tasks into the Eclipse IDE. The development of Jazz has been inspired to the Booch and Brown's vision of a "frictionless surface" for development [Booch03], which was motivated by the

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

[20] http://www-01.ibm.com/software/awdtools/rtc/

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. Jazz uses a proprietary source code management solution, which can also be replaced by other common SCM tools (e.g., SVN and Git). The Jazz client is a rich client application, called Rational Team Concert (see Fig. 9), which is built upon the Eclipse RCP platform. Other than the development-specific features, Jazz also offers a built-in RSS reader and integrates with Lotus Sametime and Google Talk instant messaging networks. Jazz repositories can also be accessed using a browser, thanks to the Jazz Rest API, which exposes and makes accessible all the core services from the Web.

GitHub[21] is a CDE service that describes itself as a "social network for programmers" (Fig. 10). Alike the other CDEs mentioned before, GitHub hosting service only offers Git as source code management to both open source and commercial software projects. However, GitHub also aims to foster developers' collaboration by letting them fork projects through Git, sending and pulling requests, and monitoring development through a twitter-like, "follow-this-project" approach. As of October 2009, GitHub community counts over 135,000 developers.



A Team Concert example project area based on the JUnit project

This example project area illustrates the use of work items, SCM, builds and other Team Concert components in a project....

It exemplary shows project work on the next JUnit release 4.4.

Jazz15111Roles determine a user's permissions as well as any preconditions and follow-up actions that are run for project and team operations. The roles assignments below are also valid in all the project's team areas. Unless configured otherwise, all users in the repository play the 'default' role.

Jazz15087Name	Jazz15088Process Roles
Bill Cassavelli	teamlead

Jazz21651No results to display

Jazz21674You may choose one of the Work Item queries below to populate this view.
More Work Item queries are available from the Work Items section of the Team Artifacts view.

Jazz21671Jazz21675Open assigned to me (JUnit Project)

Jazz21671Jazz21676Recently created (JUnit Project)

Fig. 9: A screenshot of the Jazz client Rational Team Concert

Finally, to conclude this section, we mention some other noticeable CDEs, such as Launchpad[22], well known for hosting the Ubuntu project; GNU Savannah[23], the central point for the development of most GNU software; Tigris[24] which is a CDE specialized on hosting open source software engineering tools; CodePlex[25], Microsoft's recent take on collaborative open source development.

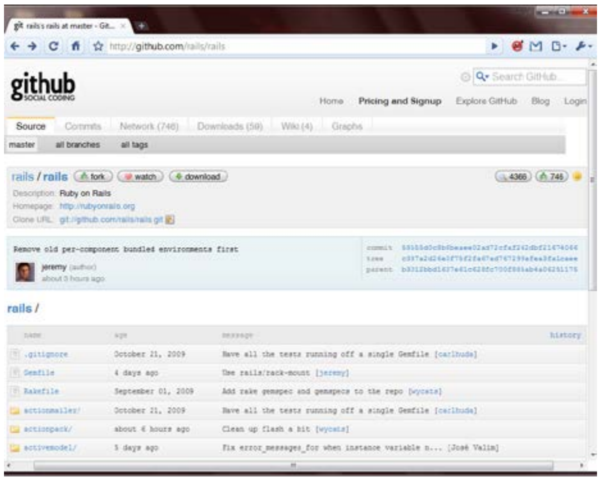
[21] http://github.com/

[22] https://launchpad.net/

[23] http://savannah.gnu.org/

[24] http://www.tigris.org/

[25] http://www.codeplex.com/



rails / rails

Description: Ruby on Rails
Homepage: http://rubyonrails.org
Clone URL: git://github.com/rails/rails.git

Remove old per-component bundled environments first
jeremy (author)
about 3 hours ago

commit [illegible]
tree [illegible]
parent [illegible]

rails /

name	age	message	history
.gitignore	October 21, 2009	Have all the tests running off a single Gemfile [carlhuda]	
Gemfile	4 days ago	Use rails/rack-mount [jeremy]	
Rakefile	September 01, 2009	Add rake gemspec and gemspecs to the repo [wycats]	
actionmailer/	October 21, 2009	Have all the tests running off a single Gemfile [carlhuda]	
actionpack/	about 6 hours ago	Clean up flash a bit [wycats]	
activemodel/	5 days ago	Fix error_messages_for when instance variable n... [José Valim]	

Fig. 10: Main page of Ruby on Rails project in GitHub

Web 2.0 extends traditional collaborative software by means of direct user contribution, rich interaction, and community building. Some key Web 2.0 applications are blogs, microblogs, wikis, social networking sites, and collaborative tagging systems. The use of Web 2.0 applications has become quite common in open source and global software projects as they represent a valuable means to increase the amount of informal communication exchanged between team members. For example, wiki platforms, such as Confluence[26], have emerged as a practical and economical option to consider for creating and maintaining group documentation [Louridas06].

Results

This section provides some anecdotes on how the most significant reviewed tools are practically used.

The idea of adopting no SCM in a distributed project is out of question. Instead, reasons in favor of selecting either a centralized or distributed code repository should be identified. In general, distributed SCM tools are the preferred choice when developers need to travel often, for example, to work remotely at customer sites, because such tools deal with merging the changes pulled from developers' repositories much better than centralized tools [O'Sullivan09]. Distributed SCM gained popularity in 2002 when Linus Torvalds took the controversial decision of using BitKeeper, a proprietary, closed source tool by BitMover Inc., for supporting the Linux kernel development, the pinnacle of free open source software. However, in 2005, when BitMover announced that it would stop providing a version of the tool free of charge to the community. Thus, Torvalds decided to start the develop a new distributed SCM, which later became Git, as none of the available free systems met his needs, particularly the requirements on performance and safeguards against data corruption, either accidental or malicious. Because distributed SCM tools have been designed with the purpose of making repositories merge a routine operation, they are in general much more performing than centralized counterparts at computing diffs and applying patches. Such difference in performance increases as the number of files in a repository reaches tens of

[26] http://www.atlassian.com/software/confluence/

thousands or more. Therefore, the adoption distributed SCM tools is highly recommended for managing very large projects.

Tracking bugs and other issues in a project is as important as code development. When Mozilla organization first came online in 1998, one of the first products that was released was Bugzilla, an open source bug system implemented to replace the in-house system then in use at Netscape. Only upon creating the bug repository, the people involved in the project could move onto the development of the new browser. Since the birth of Bugzilla, a bug is not actually a bug until it has been reported to the issue-tracking system. In fact, it is a common scenarios to forbid developers to commit any piece of code that has no issue description attached. Today, issue tracking systems have become so dependable that companies often use it also to assign and track administrative tasks.

Although all the products reviewed in this chapter are successful and effectively adopted by many distributed development teams, today companies are more and more relying on collaborative development environments. Capgemini, a multinational consultant and outsourcing company, has managed to successful introduce the use of CollabNet, the enterprise version of SourceForge, by first starting with a few pilot projects, which focused on core, most needed CDE features; then, CollabNet has been gradually spread to the various seats. Since large companies' intranets can be vast walled-gardens, hosting internal products on a common CDE has 'broken the silos', giving projects much greater visibility and fostering spontaneous collaboration across sites. Also Deutsche Bank has reported to have successfully adopted the CollabNet CDE thanks to the ability to collect all the metrics necessary to quickly target specific wastes in the project management and apply rapid corrections. At InfoSupport, a Dutch-based consultant company, the adoption of the Jazz CDE has significantly reduced maintenance costs and time to market. First, rather the spending resources in trying to make several successful tools coexist, the adoption of Jazz ensured an integrated set of tools, with a coordinated release lifecycle and no risks of present and future incompatibilities between them. Second, the availability of a web-based, thin client of Jazz allowed to give access to the relevant information within the CDE to customers.

Take-Aways

We presented a number of tools and collaborative development environments, which are available to support distributed teams. As a general guidance, we draw a few major lessons that can prevent GSE/outsourcing efforts from falling to pieces.

Since the birth of Bugzilla, a bug is not actually a bug until it has been reported to the issue-tracking system. Two aspects that drive the successful adoption of an issue tracking system are *ease of use* and *extensibility*. On the one hand. a polished, intuitive user interface lowers the entry level of expertise, thus allowing the tool to be opened to the customers as well. On the other hand, choosing products that offer extension API allows companies to customize tools to meet their corporate standards, for instance, in terms of security (e.g. single sign-on integration) or culture (e.g. polling to prioritize new features).

Wiki have mostly found their way in distributed project as document repositories and online help systems. Therefore, two aspects that drive the successful adoption of an enterprise Wiki are the strong support for *file uploading* and *WYSIWYG editing* features. In fact, on the one hand, with Wikis people found an easier way to share documents in a central place through the web browser, rather than using email or storing them in a network folder. On the other hand, Wikis have dramatically reduced the webmaster bottleneck, and the related costs, by reducing the expertise needed to update web pages, thus getting more people involved in page editing.

Communication in GSE, should be supported via a variety of different tools. This is because there is no perfect communication tool (e.g. face to face communication is easier and more comfortable than writing emails, but is volatile nonetheless). Also it should be kept in mind that many sites involved mean many different culture, habits, and, most of all, language skills, in order to avoid any ‘one tool fits all’ approach.

The idea of adopting no SCM in a distributed project is out of question. We reviewed the mainstream SCM tools, which can be broadly classified as centralized and distributed, depending on whether they need a central repository or not. Unlike centralized SCM tools, when developers check out a project from a distributed revision control system, their local machines contain a complete clone of all project’s repository (called a fork), not a just a portion of it. The major difference between a centralized and a distributed SCM tool is that with the former committing a change also implicitly means publishing it onto the central repository; conversely, with a distributed tool, commit and publish are decoupled because a developer, after committing a change to the local repository, still has to explicitly decide when to share it with others. In general, distributed SCM tools are preferred when developers need to travel often. Therefore, companies should select an SCM that reflects the degree of distribution of the project to manage. Highly distributed projects, involving three, four remote sites or more, definitely benefit from using distributed SCM. In addition, distributed SCM tools are more performing than centralized counterparts, especially for larger projects consisting of tens of thousands of files or more.

Because they are essential to enable distributed development, SCM tools were the first to be integrated within CDE products. CDEs successfully combine in one place most of the technologies mentioned earlier (e.g. issue trackers, communication and knowledge management tools) and thus provide a frictionless surface in development environments with the goal to increase the developers’ comfort and productivity. CDEs provide developers with awareness notifications, via feeds or emails, about the changes occurred to artifacts (e.g., documents being shared or modified), workspace (e.g. event notifications in case of build failures, new commits), and team (e.g. coworkers’ profiles, blogs, activities, bookmarks, wikis, and files). By aggregating this information in one place, CDEs provide an overall group awareness to developers who have little or no chances to meet, useful to speed up the establishment of organizational values, attitudes, and trust-based inter-personal connections, thus facilitating communication as well as the overall distributed software development process [Calefato09]. Although at first glance enterprise CDEs might just be discarded due to high license costs, companies should neither overlook the hidden costs due to the effort of integrating several pieces of free software, extending them to meet their corporate standards, and contacting different tech-support teams.

Finally, the area where most of the CDE platforms needs improvement is in the integration of build tools (only available in GForge, Trac, RTC, and Codeplex) and modeling tools (only available in Trac).