Methods, Tools and Environments for Collaborative Development

Alexander Filitz
07-713-928

Department of Informatics
University of Zürich
Zürich, Switzerland
alexander.filitz@uzh.ch

Abstract. Collaborative development in software engineering challenges the involved partners in different ways. This work gives a brief overview of methods, tools and environments that are used in common Global Software Development (GSD). Each of the described tools try to enhance collaboration among distributed developers. First, known classification schemes for tools in GSD are summarized. Second, one scheme is adopted and an overview on the tools related to this category is given. And the last part of this work focuses on tools that support specific processes in software engineering.

1 Introduction

The focus in this work is on methods, tools and environments, that are used in Global Software Development (GSD). In GSD, developers can be distributed all over the world working on the same project. Distribution refers to different kind of distances such as temporal, geographical and socio-cultural distance between software engineers. [39] characterise temporal distance as a cause of time zone differences or time shifting work patterns. With low temporal distance synchronous communication is much simpler to start, but management options could be reduced as well. Geographical distance is expressed as the effort that a person needs to visit another person at their site. Low geographical distance enables more co-located inter-team working, so different teams can have actual face-to-face meetings. Socio-cultural distance is measured in a directional way how one person understands another person's values and normative practices. Directional means that a person A could be socio-cultural closer to a person B than B is to A. Therefore software projects are called GSD, when they show high distance in all three mentioned ways [39].
In a concrete GSD environment, tools beeing used in every process of development must consider these special aspects and deal with them [26]. There are several benefits, described in [48], which GSD brings up:

1. Product Complexity: The complexity of a product can be distributed through various organizations.

2. Access to more work teams and skills.
3. Acquisition: Organizations can aquire chances wherever they appear.
4. Global presence and visibility of distributed organizations.
5. Reduction of time-to-market and costs.
6. Organization scale.
7. Proximity to market through distributed organizations.

In contrast to these clear benefits are challenges and problems in GSD evolving from the same distance factors mentioned above. [48] categorize them into five problem areas, namely:

1. No benefits gained by face-to-face meetings, because they are hardly possible to held.
2. The difficulty of controlling and coordinating the processes in GSD increases.
3. Through the absence of face-to-face meetings evolving problems related to trust and collaboration.
4. Different cultures and geographically distributed teams result in loss of team spirit which is gained through co-located teams.
5. Missunderstandings can result in a bad performance of a team.

Section two is about different classification schemes that can be used to categorize methods, tools and environments in the context of GSD and collaboration. Section three briefly describes tools related to model-based collaboration and section four addresses tools for specific processes in the software lifecycle.

2 Classification Schemes for Tools

The search for classification schemes did not produce a clear result. [48] have chosen the ISO/IEC 12207[1] for software lifecycle processes. This standard defines three types of processes: primary, supporting and organizational processes which can be grouped to software specific process and system context processes. After collecting a survey about tools being used by multinational companies, they grouped these tools into following three groups related to the ISO/IEC 12207 standard:

1. Tools supporting project processes.
2. Tools supporting software implementation processes.
3. Tools supporting software support processes.

Project processes basically support the project management activities. The second group includes processes like requirements analysis, architectural design, detailed design, construction and integration. Support processes include the documentation, configuration management and review and quality assurance processes.

[1] `http://www.iso.org/iso/catalogue_detail.htm?csnumber=43447`

Another approach has its origin in the nature of software engineering collaboration itself. "Software engineering collaboration can thus be understood as artifact-based, or model-based collaboration, where the focus of activity is on the production of new models, the creation of shared meaning around the models, and elimination of error and ambiguity within the models [64]". According to this point of view, [64] defined four categories for tools used in GSD:

1. Model-based collaboration tools
2. Process support tools
3. Awareness tools
4. Collaboration infrastructure

The first group enables developers to collaborate in different models or representations of the software, for example data models or UML diagrams. Process support tools could be tools which implement specific work processes like Subclipse [58] or most of the available Software Configuration Management (SCM) tools. Awareness tools are used to e.g. show developers the current activity of an artifact and who is working on it preventing possible conflicts. Collaboration infrastructures try to enhance the compatibility between different collaboration tools with a focus on data and control integration [64].
A third framework was introduced by [41] which describes these four different groups:

1. Development tools
2. Integrated development environments
3. Software-as-a-service
4. Consulting & services

Development tools assist a specific phase or task of the software process. For example a document management system, testing tools, quality assessment, performance monitoring and source code management. Further these tools can be divided into single task tools and single process tools, where single process tools enhance a distributed software project and single task tools do not support GSD.
Integrated Development Environments (IDE) can be divided into three types:
Stand-alone IDE's optimized for a single programmer, enhanced IDE's offering team features like bug tracking and collaborative IDE's which support collaboration among software engineers through special functionalities. The latter are tradionally implemented as client/server applications. The third group Software-as-a-Service (SaaS) sees software functionality more as a set of distributed services that can be coupled and configured at delivery time. In general this leads to a centralized development environment for distributed teams. Tools in the fourth category consulting and services support the ongoing development processes that can be done by external service providers and consultancies. But this approach is not an effective solution for GSD [41].
The next section describes tools related to model-based collaboration reaching from requirement analysis to testing and inspection. The first group from the classification scheme developed by [64] is used, because most of the tools, methods and environments found during research fit in this group.

3 Model-based Collaboration

3.1 Requirement Analysis

There exist a lot of tools supporting the requirement analysis. Because this work focuses on tools related to GSD, it does not include tools whithout support for collaboration among software engineers. The tool eRequirements [29] is a step-by-step web application for managing requirements. After finishing all steps, it is possible to automatically generate a requirement specification as PDF. Another web application is Gatherspace [30] which supports use case creation as well automated report generation. Additionally to the other tools test case and bug tracking management is included and can be linked to defined requirements.
Borland CaliberRM [12] uses a central repository to store the requirements and supports various clients such as web browser, Eclipse, Microsoft Visual Studio and Microsoft Windows. It supports traceability of requirements to artifacts and it is possible to visualize that.
IBM Rational DOORS [25] uses a central repository as well to make the requirements accessible to the hole team, inlcuding support of simultaneous editing and viewing of documents. So-called "power users" access the repository through a desktop client software with full editing support and other stakeholders access trough standard web browsers. By supporting the requirement interchange format suppliers and other partners can contribute their own requirements and so are fully integrated in the hole process.
IBM RequisitePro [52] is accessed through a web interface. It uses the Microsoft Word document to edit and communicate requirements. The document is connected via a plugin to the central database and it is also possible to access the database without this feature. IBM Requirements Composer [20] is part of the IBM Jazz platform and uses as well a web interface for the starting point in collaboration. It supports the creation of use case diagrams and includes a report wizard guide to automatically generate your document in many different file formats. IBM Requirements Composer integrates well with the DOORS and the RequisitePro platform. In [22] you can find a nice video that shows how a requirement can be transfered from one to another platform.
RavenFlow [51] features two clients, the express client as a Microsoft Office add-in product, or the professional desktop client. Collaboration is supported through a central server and a checkout/checkin system [64]. As we can see collaboration is often enabled through a central repository or database in interplay with a web interface.
Current research is active in the process of Requirements Elicitation (RE), because RE is crucial for the failure prevention in software projects [33]. In this phase all stakeholders are involved in the discovery of requirements, which is especially difficult in a GSD context. There is a lack of RE tools with special support to traditional RE methods e.g. brainstorming and workshops [57]. Thus [57] developed a spatial hyptertex wiki (ShyWiki) that supports RE methods based on the KJ method [36], Nominal Group Technique [24] and EasyWinWin [27]. The process starts with defining the initial stakeholders and requirement cat-

egories thereafter one or more brainstorming sessions for each category is necessary. The next step is the grouping phase, where redundant information is deleted. This phase is followed by a priorization voting with values between 1 and 10 on every requirement. The last step is called refinement and includes the actual requirements modeling [57].
The field of other wikis related to requirements elicitation includes WikiWinWin [65], SOP-wiki [23], SmartWiki [37] and SWORE [53]. But all of them have no sufficient support of brainstorming. For example WikiWinWin is based on brainstorming but in fact does not support any collaborative interpretation [57].
ART-SCENE [6] is a use case based RE tool that is able to automatically generate scenarios for further processing. Ongoing work has developed a PDA plugin to ellaborate new use cases and requirements on site [64] [6]. EGRET [56] is an Eclipse plugin developed for requirements analysis which uses separated repositories for different types of stakeholders.
These mentioned tools should not be seen as a complete list of available tools for requirements analysis rather as overview on what is actually being used or being researched. But in the end all of these collaborative tools can only achieve good results if a culture of collaboration in the project is established [56]. In table 1 you can find a short summary of the described tools above.

Table 1. Tools for requirements analysis

Name	Group	Availability
eRequirements	Web Application	Free
Gatherspace	Web Application	Commercial
Borland CaliberRm	Desktop	Commercial
RavenFlow	Desktop	Commercial
IBM Rational DOORS	Desktop & Web Interface	Commercial
IBM RequisitePro	Web Interface	Commercial
IBM Requirements Composer	Web Interface	Commercial
ART-SCENE	Web Interface	Not public
ShyWiki	Wiki	Not public
WikiWinWin	Wiki	Not public
SOP-wiki	Wiki	Not public
SmartWiki	Wiki	Not public
EGRET	Eclipse Plugin	Not public

3.2 Architecture Design

Collaboration in architecture design of a software project includes activities that are not part in the tools focused on architecture design. There is a lot of negotiating between different models and often political decisions are made. Let's

start with yet another IBM tool, the Rational Software Architect [4]. This tool is built on the Eclipse software framework and can be used as single instance as well as an Eclipse plugin. It supports modeling with UML and forward, reverse transformations from model to code. Collaboration is offered through a configuration management system that helps to develop the models in parallel. An almost similar tool is part of the Borland product family, Borland Together [63]. Together also uses the Eclipse technology.
The award-winning Enterprise Architect [3] supports, among other things, full traceability from requirements to deployment, a powerful document generator and collaboration through a scalable, team-based repository. ArchStudio [5] is an Eclipse plugin developed by the University of California, Irvine that is based on the xADL 2.0[2] architecture description language which enables architecture meta-modeling as well. It supports collaboration by versioning the architecture description files [64]. The same approach of collaboration uses ACMEStudio [1] which is implemented as an Eclipse plugin. MolhadoArch [43], developed by the Iowa State University, uses a much more complex version control system where it is possible to collaborate on the level of single architectural elements.

Recent research studied the modeling and management of Architecture Knowledge (AK). [60] compared some of the methods, models and tools which have been proposed in the last few years: Architecture Design Decision Support System (ADDSS) [13] is a web application which supports full traceability through showing links between requirements and design decisions. It also supports automated report generation of the architecture and the decisions behind it and is able to visualize the growth of AK over time [60]. The tool Knowledge Architect [38] makes use of three different clients, the first is a Microsoft Word plugin for managing AK, the second for quantitative architectural analysis models as a Microsoft Excel plugin and the third features as an explorer to visualize and analyze the AK [60]. The central repository server enables collaboration among distributed software architects. Another web application is the Process-centric Architecture Knowledge Management Environment (PAKME) [8] which implemented a data model to describe design decisions, alternatives, rationale and quality attributes [48] [60]. Table 2 shows the mentioned architectural design tools.

3.3 Detailed Design

Nowadays, UML is the most accepted and distributed modeling standard. This leads to the fact that most fine design tools are synonymous with UML editors. We have seen that collaboration in architectural design tools mostly depend on an underlying SCM [64]. Realtime distributed modeling and collaboration other than discussion and SCM though is not well supported in actual software. But research has been active and proposed some tools related to distributed modeling. The research tool called CAMEL [14] supports multiple diagram types as

[2] `http://www.isr.uci.edu/projects/xarchuci/`

Table 2. Tools for architectural design

Name	Group	Availability
IBM Rational Software Architect	Eclipse Plugin	Commercial
ArchStudio	Eclipse Plugin	Free
ACMEStudio	Eclipse Plugin	Free
Borland Together	Desktop	Commercial
Enterprise Architect	Desktop	Commercial
MolhadoArch	Desktop	Not public
Knowledge Architect	Desktop	Not public
ADDSS	Web Application	Free
PAKME	Web Application	Not public

well as drawing sketches. The focus is on managing the design meetings and capturing all the relevant information in a global accessible repository.
GroupUML [9] was developed as a desktop application and has its origin in research too. The tool supports distributed modeling with UML and includes a chat environment for discussion. Sysiphus [10] has a lot in common with GroupUML. The initial prototype was a desktop rich client and developed for research. It also provides special traceability and awareness mechanisms. A different approach was researched by [15] and their tool SUMLOW. This supports same-time, same-place UML diagram creation via a shared electronic whiteboard [64].
Gliffy [32] is one of the tools that allows collaboration among detailed design activities. The primary tool is a web application, but it is also possible to use it as a JIRA [34] plugin. Team-members must be invited to a session so they can work together on the diagram. Gliffy supports email notifications and a lot of diagram types additional to UML including flowcharts, SWOT and venn diagrams.
IBM's Rational product family includes Tau [61] which is based on the UML 2.1 standard. Rational Tau includes a automatic document generation, design simulations and an automated test management. It integrates well with Rational System Architect [48].
Again this summary focused on tools which enable collaboration between distributed software engineers. For example a complete list of available UML tools can be found here [57]. Table 3 summarizes the tools for detailed design. Some architectural design tools like Enterprise Architect or Borland Together are not specificely included in this table, although they support detailed design of software artifacts.

3.4 Testing and Inspections

Testing and requirements analysis have several things in common, e.g. it is an iterative process with a lot of communication involved. This leads to an inten-

Table 3. Tools for detailed design

Name	Group	Availability
CAMEL	Desktop	Not public
GroupUML	Desktop	Not public
Sysiphus	Desktop	Not public
SUMLOW	Desktop	Not public
Gliffy	Web Application & JIRA Plugin	Commercial
IBM Rational Tau	Desktop	Commercial

sive collaboration between stakeholders and software engineers. It is common to offer beta version testing, where new users report bugs and ambiguities to the development team. This interface is mainly managed through a bug tracking system [64]. These systems support the recording of initial error reports, priorization of errors and comments on error reports. They are usually accessible trough common internet browsers. For example BugZilla [11] is a very popular bug tracking system that is used e.g. by NASA and facebook. A huge comparison list of such systems can be found here [45].
Distributed testing is e.g. supported by the web application TestLink [62] and the Firefox Addon Selenium [54]. TestLink offers a hole test creating and tracking management environment including requirements traceability and report generation. Selenium allows you to write and run actual tests on the web browser in many different languages. There is also a possibility to distribute the execution of a test case to multiple servers, to run tests on a lot of platforms at the same time.
In the process of reviewing, code inspections have a long tradition because they can be viewed as the best-practice to improve software quality [48]. The Internet-Based Inspection System (IBIS) [59] is a tool for distributed inspection teams. Its goal is to minimize synchronous activities and coordination problems. IBIS was implemented as a web application and supports email notification as well as forum discussions [48].
An older web-based solution is Hypercode [47] that allows asynchronous code inspections for distributed teams and supports email notifications as well. The IBM tool Rational Quality Manager [40] supports lifecycle quality management from requirements to test cases and error handling. Collaboration is enabled through a web 2.0 interface and scales well in large team structures.
The Eclipse plugin Jupiter [35] is a lightweight and flexible review tool, its goal is to be more efficient at collecting and analyzing code than handmade approaches. Table 4 summarizes the tools for testing and inspections.

Table 4. Tools for testing and inspections

Name	Group	Availability
BugZilla	Web Application	Free
TestLink	Web Application	Free
Selenium	Firefox Addon	Free
IBIS	Web Application	Free
Hypercode	Web Application	Not public
IBM Rational Quality Manager	Desktop & Web Interface	Commercial
Jupiter	Eclipse Plugin	Free

4 Process Support Tools

4.1 Tools Supporting Project Processes

GSD or large software projects in general need to follow a clear structure defining roles for engineers and a sequence of steps for creating the artifacts. After getting familiar with an organization specific structure for software projects, engineers are able to reduce coordination activities among team members [64].
In activeCollab [2] project work is organized according to pre defined milestones. It supports planning of activites and tracking of time as well as email integration and automated invoicing. The tool is implemented as a web application and installable on your own web server.
Assembla [7] is another web application which key features are an activity stream of all projects members and integration of many services such as ticketing, chat, wiki and time tracking. Collaboration is supported through the use of these web based services and features [48].
Apache developed a build system called Maven [42] that can be used as an Eclipse plugin as well. It supports automated build processes and integration with plugins e.g. to generate automated source documentation or to verify test cases. Rational Team Concert [21] from IBM offers a collaborative development environment for agile and traditional methods. The tool includes reporting, work item management, source & change management and build management and it is part of the IBM Jazz platform. IBM Rational Method Composer [19] is more a process management platform with a large process library to support the creation and implementation of processes in software projects.
Microsoft Sharepoint [55] can be used as well for collaborative project management. It enables its users to easily create web sites to share information and manage documents and features different kind of communities for example wikis, ratings or organization browser. Microsoft Project [50] enables unified project and portofolio management, includes time and business reporting, visual team planner and integrates well with the Sharepoint solution. Jira [34] is a powerful tool that comes from a high level of integration through plugins. It supports bug tracking, agile project development, custombisable workflows and reporting.

Nevertheless still office applications are being used such as Microsoft Office or OpenOffice despite the lack of collaboration possibilities [49]. A wide list and comparison of project management tools can be found here [46].

Table 5. Tools supporting project processes

Name	Group	Availability
activeCollab	Web Application	Commercial
Assembla	Web Application	Commercial
Maven	Desktop & Eclipse Plugin	Free
IBM Rational Team Concert	Desktop	Commercial
IBM Rational Method Composer	Desktop	Commercial
Microsoft Sharepoint	Desktop & Web Application	Commercial
Microsoft Project	Desktop	Commercial
Jira	Desktop	Commercial

4.2 Tools Supporting Construction Processes

Normally there is no concrete collaboration in the act of programming new pieces of code. But agile software development has shown that for example pair programing can improve the quality of a software project. [48] have collected the following tools which add new features of collaboration in the construction process.
CollabVS [18] adds real-time collaboration features to the Microsoft VisualStudio platform. It can display information about online team members and which part of the code is actually being edited. For example there is a possibility for collaborative code reviewing without leaving the IDE. Pair programming is supported by the research tool COPPER [44]. In this tool one can see which part of the code is edited or viewed by other team members. It also implements features of groupware systems like communication tools for example instant messaging.
Cola [17] which is based on the Eclipse Communication Framework(ECF) supports shared and synchronous editing of source code. Here [28] you can see a nice video how this works. GForge [31] is a web application written in PHP with a huge toolset and features collaborative source code management, trackers, task managers, document managers and forums.
Google Code [16] supports, beside being a central code repository, a lot of tools such as on the web compilers, debuggers and API's to help developers creating their software. This web application is free to use for everybody.

5 Conclusions

A lot of tools and environments offer different kind of collaboration in development for software engineers, but in the context of GSD there is still need of

Table 6. Tools supporting construction processes

Name	Group	Availability
CollabVS	Desktop	Not public
COPPER	Desktop	Not public
Cola	Eclipse Plugin	Free
GForge	Web Application	Commercial
Google Code	Web Application	Commercial

more improvement and research. For example traditional desktop IDE's could be better integrated with web-based solutions and support for multi-project collaboration is an upcoming problem in large systems-of-systems [64].
As we have seen the majority of the described tools are offering a web based environment for collaboration. This is possible by new technologies such as AJAX[3] which fulfill the earlier lack of user interface interactivity. Despite the growing acceptance for web based solutions there are the longstanding desktop IDE's which will not disappear or being replaced in the future. There will be more a mixture between web based and desktop tools. For example code editing is more intuitive a desktop concern and the collaboration is handled pretty well by common SCM systems [64].
Tools offering a web interface for collaboration need to have a possibility to work in an offline mode. So collaboration is secured even if the central web server is not working. Most of the tools include asynchronous communication like email notifications and threaded forums, but tools with synchronous activities lack of synchronous communication e.g. both architectural and fine design tools could make use of this communication since this is natural in co-located development [48].
Social-networks and user profile integration, that e.g. can be found in GoogleCode, can be used to reduce the cultural differences and its impact. [48] says that this could be an open gap for researchers and companies to invent more socio-cultural features for collaboration in development.
Last but not least the interconnection and integration between tools, supporting different phases of the software lifecycle, is really low. Something like the perfect "power-tool" supporting all the different processes does not exist and it is difficult to select appropriate tools for each development phase. The IBM Rational Jazz[4] platform tries to connect and integrate the included tools, but this leads to proprietary data models which could have an effect on sustainable collaboration. That is the reason why [48] advises to use solutions like Eclipse and their plugin system to avoid single vendor-driven tool integration and their possible bad impact on successfull collaboration [48].

[3] `http://en.wikipedia.org/wiki/Ajax_(programming)`
[4] `http://www-01.ibm.com/software/rational/jazz/`

References

1. ACMEStudio. http://www.cs.cmu.edu/~acme/AcmeStudio/index.html.
2. activeCollab. http://www.activecollab.com/.
3. Enterprise Architect. http://www.sparxsystems.com.au/products/ea/index.html.
4. IBM Rational Software Architect. http://www-142.ibm.com/software/products/de/de/swarchitect-websphere/.
5. ArchStudio. http://www.isr.uci.edu/projects/archstudio/.
6. ART-SCENE. http://hcid.soi.city.ac.uk/research/Artsceneindex.html.
7. Assembla. http://www.assembla.com/.
8. M.A. Babar, X. Wang, and I. Gorton. Pakme: A tool for capturing and using architecture design knowledge. In *9th International Multitopic Conference, IEEE INMIC 2005*, pages 1–6, December 2005.
9. N. Boulila. Group support for distributed collaborative concurrent software modeling. In *Automated Software Engineering, 2004. Proceedings. 19th International Conference on*, September 2004.
10. Bernd Bruegge, Allen H. Dutoit, and Timo Wolf. Sysiphus: Enabling informal collaboration in global software development. In *Global Software Engineering, 2006. ICGSE '06. International Conference on*, October 2006.
11. BugZilla. http://www.bugzilla.org/.
12. Borland CaliberRM. http://www.borland.com/us/products/caliber/index.html.
13. Rafael Capilla, Francisco Nava, Sandra Pérez, and Juan C. Due nas. A web-based tool for managing architectural design decisions. *SIGSOFT Softw. Eng. Notes*, 31(5), 2006.
14. M. Cataldo, C. Shelton, Yongjoon Choi, Yun-Yin Huang, V. Ramesh, D. Saini, and Liang-Yun Wang. Camel: A tool for collaborative distributed software design. In *Global Software Engineering, 2009. ICGSE 2009. Fourth IEEE International Conference on*, July 2009.
15. Qi Chen, J. Grundy, and J. Hosking. An e-whiteboard application to support early design-stage sketching of uml diagrams. In *Human Centric Computing Languages and Environments, 2003. Proceedings. 2003 IEEE Symposium on*, October 2003.
16. Google Code. http://code.google.com/.
17. Cola. http://code.google.com/p/cola-rtse/.
18. CollabVs. http://research.microsoft.com/en-us/projects/collabvs/.
19. IBM Rational Method Composer. http://www-01.ibm.com/software/awdtools/rmc/index.html.
20. IBM Requirements Composer. http://www-01.ibm.com/software/awdtools/rrc/.
21. IBM Rational Team Concert. http://www-01.ibm.com/software/awdtools/rtc/.
22. Video creating a requirement in Rational Requirements Composer and transferring it to RequisitePro for management. http://www.ibm.com/developerworks/offers/lp/demos/summary/r-rrcreqpro.html.
23. B. Decker, E. Ras, J. Rech, P. Jaubert, and M. Rieth. Wiki-based stakeholder participation in requirements engineering. *Software, IEEE*, 24(2):28–35, March-April 2007.
24. A. Delbecq and A. Van de Ven. A group process model for problem identification and program planning. *Journal of Applied Behavioral Science*, 18(2):466–492, 1971.
25. IBM Rational DOORS. http://www-01.ibm.com/software/awdtools/doors/.

26. Kevin Dullemond and Ben van Gameren. Technological support for distributed agile development. Master's thesis, Delf University of Technology, 2009.
27. EasyWinWin. http://csse.usc.edu/csse/research/easy_win_win/.
28. Cola: Real-Time Shared Editing. http://www.vimeo.com/1195398?pg=embed\&sec=1195398.
29. eRequirements. http://www.erequirements.com/app.
30. Gatherspace. http://www.gatherspace.com/index.html.
31. GForge. http://gforgegroup.com/index.php.
32. Gliffy. http://www.gliffy.com/.
33. H.F. Hofmann and F. Lehner. Requirements engineering as a success factor in software projects. *Software, IEEE*, pages 58–66, 2001.
34. JIRA. http://www.atlassian.com/software/jira/.
35. Jupiter. http://code.google.com/p/jupiter-eclipse-plugin/.
36. J. Kawakita. The original kj-method. Technical report, Kawakita Research Institute, Tokyo, 1982.
37. E. Knauss, O. Brill, I. Kitzmann, and T. Flohr. Smartwiki: Support for high-quality requirements engineering in a collaborative setting. In *Wikis for Software Engineering, 2009. WIKIS4SE '09. ICSE Workshop on*, 2009.
38. Peng Liang, Anton Jansen, and Paris Avgeriou. Knowledge architect: A tool suite for managing software architecture knowledge. Technical report, University of Groningen, Software Engineering and Architecture (SEARCH) Group, 2009.
39. Brian Lings, Bjrn Lundell, Pr J. Agerfalk, and Brian Fitzgerald. A reference model for successful distributed development of software systems. In *International Conference on Global Software Engineering*. IEEE, 2007.
40. IBM Rational Quality Manager. http://sourceforge.net/projects/ibis/.
41. R. Martignoni. Global sourcing of software development - a review of tools and services. In *Global Software Engineering, 2009. ICGSE 2009. Fourth IEEE International Conference on*, July 2009.
42. Maven. http://maven.apache.org/.
43. MolhadoArch. http://home.eng.iastate.edu/~tien/molhado/molhado_arch.html.
44. H. Natsu, J. Favela, A.L. Moran, D. Decouchant, and A.M. Martinez-Enriquez. Distributed pair programming on the web. In *Computer Science, 2003. ENC 2003. Proceedings of the Fourth Mexican International Conference on*, September 2003.
45. Comparison of issue-tracking systems. http://en.wikipedia.org/wiki/Comparison_of_issue_tracking_systems.
46. Comparison of project management software. http://en.wikipedia.org/wiki/Comparison_of_project_management_software.
47. D.E. Perry, A. Porter, M.W. Wade, L.G. Votta, and J. Perpich. Reducing inspection interval in large-scale software development. *Software Engineering, IEEE Transactions on*, 28(7):695 – 705, July 2002.
48. Javier Portillo-Rodriguez, Aurora Vizcaino, Christof Ebert, and Mario Piattini. Tools to support global software development processes: A survey. In *International Conference on Global Software Engineering*. IEEE, 2010.
49. C.R. Prause, R. Reiners, and S. Dencheva. Empirical study of tool support in highly distributed research projects. In *Global Software Engineering (ICGSE), 2010 5th IEEE International Conference on*, 2010.
50. Microsoft Project. http://www.microsoft.com/project/en/us/default.aspx.
51. RavenFlow. http://www.ravenflow.com/products/.
52. IBM Rational RequisitePro. http://www-01.ibm.com/software/awdtools/reqpro/.

53. T. Riechert and T. Berger. Leveraging semantic data wikis for distributed requirements elicitation. In *Wikis for Software Engineering, 2009. WIKIS4SE '09. ICSE Workshop on*, 2009.
54. Selenium. `http://seleniumhq.org/`.
55. Microsoft Sharepoint. `http://sharepoint.microsoft.com/en-us/Pages/default.aspx`.
56. Vibha Sinha, Bikram Sengupta, and Satish Chandra. Enabling collaboration in distributed requirements management. *Software, IEEE*, 23(5):52–61, September-October 2006.
57. Carlos Solis and Nour Ali. Distributed requirements elicitation using a spatial hypertext wiki. In *Global Software Engineering (ICGSE), 2010 5th IEEE International Conference on*, pages 237–246, 2010.
58. Subclipse. `http://subclipse.tigris.org/`.
59. Internet-Based Inspection System. `http://sourceforge.net/projects/ibis/`.
60. Antony Tang, Paris Avgeriou, Anton Jansen, Rafael Capilla, and Muhammad Ali Babar. A comparative study of architecture knowledge management tools. *Journal of Systems and Software*, 83(3):352 – 370, 2010.
61. IBM Rational Tau. `http://www-01.ibm.com/software/awdtools/tau/`.
62. TestLink. `http://www.teamst.org/`.
63. Borland Together. `http://www.borland.com/us/products/together/index.html`.
64. Jim Whitehead. Collaboration in software engineering: A roadmap. In *International Conference on Software Engineering*. IEEE, 2007.
65. Da Yang, Di Wu, S. Koolmanojwong, A. Winsor Brown, and B.W. Boehm. Wikiwinwin: A wiki based system for collaborative requirements negotiation. In *Hawaii International Conference on System Sciences, Proceedings of the 41st Annual*, 2008.