

# Supporting Teams with Open Source Software Tools

Clifton Kussmaul, Muhlenberg College

## Abstract

Student teams and projects present many challenges. For example, students need to know what to do, and benefit from seeing “real world” tools and processes. Teachers need to assess progress and identify potential problems. Some of these challenges can be addressed using open source software tools. These tools are free and have often been adapted to a variety of settings, and faculty and students with software development skills can examine and enhance the tools.

This paper describes experiences in academia and industry using such tools, including version control systems, wikis, task trackers, and integrated systems such as Trac and Moodle. The paper describes how such tools can address common challenges, as well as lessons learned and recommendations for selecting and introducing such tools effectively. The paper emphasizes the benefits of focusing on people and processes, choosing a small number of flexible tools, introducing them incrementally, and accepting occasional failures.

## Introduction

Team projects have a long history in education, and there is an extensive literature on the benefits, risks, and best practices for managing team projects (e.g., Fincher, Petre, and Clark 2001; Tomayko 1987; Tomayko 1998). This paper focuses on how open source software tools can support team projects by helping to address several common challenges:

- How to encourage teams to focus on essential activities and not get distracted?
- How to help students appreciate how and when specific tools and structures are appropriate, so students use them thoughtfully rather than blindly?
- How to assess team and student performance, and the success of the course or individual activities?

## Open Source Software

Open source software generally refers to software that is distributed without charge and with the original source code, so that other software developers can fix defects, add enhancements, or otherwise modify the software and share their changes with others. There are a large, and at times overwhelming, number of open source software projects; as of October 2007, SourceForge.net hosted over 150,000 open source software projects, although many of them are small and have little activity.

Open source software offers both benefits and risks. It can be freely installed on any number of computers. It can be modified by faculty, staff, and students with appropriate knowledge. It can be more difficult to evaluate than commercial alternatives, which place more emphasis on sales and marketing. It may include less documentation and require more expertise to install and maintain. When evaluating open source software, consider a variety of factors:

- Are new versions released at least once a year? Realize that newer projects often release more frequently than more mature projects.
- How many people are actively involved in development? While most open source projects have a few people who do most of the work, a project with only one key developer could stall if that person stops work.
- How many people use the project and contribute to online discussions about it?
- What support options are available? Many open source projects offer commercial support or consulting assistance.

## Taxonomies & Objectives

Team activities and supporting tools can be categorized in several ways:

1. Whether they occur at the same time (synchronous) or at different times (asynchronous).
2. Their supported media type(s), e.g., text, auditory, and/or visual.
3. Their objective(s): cohesion, communication, collaboration, or coordination.

**Cohesion.** Team members need to develop and maintain a sense of community, trust, and shared purpose. Tools can only indirectly support cohesion, although it is perhaps the single most important objective.

**Communication.** Team members need to share information with each other. This can be done in person in a small, collocated team, but appropriate tools can be invaluable, particularly for teams that span physical locations, time zones, and organizational or political boundaries. Such virtual teams are increasingly common. Communication tools are commonplace, familiar to most students and teachers, and either free or inexpensive, so this paper will not focus on them.

**Collaboration.** Team members need to work together on a variety of documents. Such collaboration can be done informally for small, simple projects, but larger projects require more planning and organization, which can be facilitated by appropriate tools. Such tools can help ensure that everyone has access to the most current version of a document, prevent or help to resolve conflicts when multiple people try to edit the same document, keep track of changes during the life of a document, and provide access to previous versions.

**Coordination.** Team members need to know what tasks need to be performed, how important they are, who is working on which tasks, what resources (people, time, money) each task will take, and when each task must be completed. Again, appropriate tools can facilitate such coordination, particularly for larger, more complex projects.

Each of these four objectives becomes more difficult and more critical for larger teams and projects. For example, there is good evidence that software development productivity declines dramatically for large projects (Jones 2000, 65).

### Collaboration: Sharing Documents and Information

A simple but awkward way to collaborate is to transfer documents using physical media, such as floppy disks or USB drives. It is somewhat better to email documents between team members, or save documents on a shared file server, so that there are multiple copies of the document, and some hope of retrieving an older version if something goes wrong. However, these approaches are insufficient for projects with many documents and people.

Software developers have faced such challenges for decades, and have developed version control systems (VCS). VCSs typically save all versions of every document, and log the date, author, and a description of each new version. This makes it easy to retrieve previous versions and to review the history and evolution of a document. They provide ways to lock documents so that other people can't edit them, as well as ways to merge different versions. There are many open source and proprietary VCSs, but the most common open source systems are Concurrent Versions System (CVS) and SubVersion (SVN). Although VCSs are usually designed to work with plain text files, they can also handle other file types. For example, I use a SVN repository for all of my software code, web pages, academic papers, teaching notes, examples, and handouts. A number of authors discuss ways to incorporate VCSs in courses and extract useful data from them (e.g., Clifton, Kaczmarczyk, and Mrozek 2007; Glassy 2006; Hartness 2006; Liu, Wong, and German 2004; Mierle, et al 2005; Reid and Wilson 2005).

A wiki provides many of the benefits of a VCS in a form that is easier for non-programmers to use. A wiki is a special website that allows any user to create and edit pages, and link pages together, without specialized tools or knowledge (Lamb 2004; Leuf and Cunningham 2001). There are many different wiki systems, each with strengths and weaknesses (e.g., CosmoCode 2007; WikiEngineComparison 2007; Schwartz, Clark, Cossarin, and Rudolph 2004). Typically, pages are edited as plain text with simple formatting conventions, although some wikis include graphical text editors. This makes it easy to highlight key ideas or comment on other people's work. Wikis also allow users to see a page's history, including all previous versions, and which changes were made by whom. The history also allows students and teachers to review and reflect on how documents evolve over time. Several authors review

ways to use wikis for team projects and to analyze outcomes (e.g., Kay, et al 2006; Korfiatis and Naeve 2005; Viégas, Wattenberg, and McKeon 2007).

The best known wiki is probably Wikipedia, a free-content encyclopedia with over five million articles in over 250 languages. However, wikis can also be used in other ways for a variety of creative and educational purposes (e.g., Angeles 2004; Fountain, 2007; Kussmaul, Howe, and Priest 2006; Kussmaul and Albert 2007; Notari 2006; Read 2005; Wei, Maust, Barrick, Cuddihy, and Spyridakis 2005). I have direct experience with several wiki systems. TWiki is widely used in corporate intranets, and has a wide variety of plugins to add specialized functionality, but may be too complex for simple projects. MoinMoin and PmWiki are easier to install and configure, and are flexible enough for a variety of smaller projects. MediaWiki is used for Wikipedia, but I have found it more difficult to adapt to other purposes. The Moodle course management system and the Trac project management system both contain wikis that lack some common capabilities, but contain some special features; for example, the Moodle wiki can easily be configured to support individual, group, or whole class assignments, while the Trac wiki can automatically link to the VCS and the task tracking system (see below).

Finally, document management systems (DMS) are also designed to help teams manage documents. A DMS is particularly useful for workflow, where documents go through specific revision and approval stages. Again, there are numerous proprietary and open source DMSs. I have not experimented with using a DMS for team projects, however.

### Coordination: Tracking Tasks

For small, co-located teams with relatively simple projects, the team members may be able to coordinate their activities informally. Although this seems simple and straightforward, it can still lead to confusion, and does not scale well to larger projects.

A better approach is to maintain a shared document that tracks every task and its current status. This can be a very important document, so it should be stored on a shared file server, under version control, or as part of a wiki (see above). Although a plain text file or a word processor document can work well, I recommend using a spreadsheet, with a row for each task and a column for each piece of associated information, such as priority, difficulty, current owner, start date, completion date, etc. This makes it easy to sort or filter tasks, and to do simple reporting, such as total effort, or the number of tasks for each team member.

For larger, more complex projects, more sophisticated task tracking tools (both proprietary and open source) are available. Bugzilla is a well known open source task tracking system, intended specifically for defect tracking. The Trac project management system includes a task tracking system that works well for both tasks and defect reports. Task trackers can also provide useful data about team behavior (e.g., Liu 2005).

### Putting Them Together

The previous section described how open source tools could be used to support specific team activities. However, teams usually perform a variety of activities, which could benefit from different tools. Teams can approach this situation in two ways. First, they can use a tool that provides multiple functions and try to adapt it to multiple activities. Second, they can use multiple tools and try to make the tools work well together.

### Multifunction Tools

Tools that provide multiple functions are often the easiest to use, since the functions were designed to work together. For example, course management systems such as Moodle (open source) and Blackboard (commercial) provide a variety of functions designed to work smoothly together, and include functions that can support team projects; this can be a good first approach for institutions that already have such a system. I have used Moodle to support courses, college committees, and research collaborations. Similarly, the Trac project management system combines SVN, a simple wiki, and good task tracking capabilities. I have used Trac successfully for industry projects and student course projects. There are a variety of other groupware systems designed to support communication, collaboration, and coordination.

Although multifunction tools can be more difficult to install and maintain than individual projects, the total effort is usually less than the effort required to install, integrate, and maintain individual projects. Another

advantage is that there may be commercial hosting providers (this is true for both Moodle and Trac). A disadvantage of such tools is that their individual functions may be less sophisticated than in more narrowly specialized tools. For example, the wiki contained in the Trac project management system lacks features common to most other wikis, such as the ability to rename pages.

### Multiple Tools

Tools designed for a specific function are often simpler and more effective to use, and are usually easier to install, maintain, and modify. Thus, a team or teacher can choose the set of “best of breed” tools that best fit the current requirements and constraints.

However, using and integrating multiple tools presents a number of challenges, particularly when three or more tools are involved. The team must learn to use each tool, and agree on which tool to use for which activities, particularly when the tools have overlapping capabilities. For example, are meetings scheduled via email, tickets, or a wiki page? What about customer requests? In 2006 my software engineering project course used the discussion forums and grading capabilities in Moodle, and the VCS, wiki, and task tracking in Trac. Supporting materials ended up scattered across both systems, and no one was sure where to find them. In 2007 we used Trac for everything; it didn’t do everything perfectly, but there was less confusion. Installing and maintaining multiple tools takes more effort, particularly if the tools need to share information (such as user names and passwords). It also requires more experience to evaluate, select, and integrate combinations of tools.

Sometimes, however, multiple tools can work well together. For example, Bugzilla and MediaWiki are more powerful and flexible than the corresponding functions within Trac, and there are detailed instructions for configuring Bugzilla, SubVersion, and MediaWiki to create a more complete development environment (Segetech Ltd. 2007). These types of integrated systems will become more common, as open source projects compete to provide more functionality with less effort.

### Student Survey Results

During Fall 2006, the seven students in a software engineering project course were surveyed regarding the usefulness of the Trac project management system. Their quantitative responses are summarized below.

1=very negative, 5 = very positive

Overall	4.1
To store & share info	4.1
To communicate with team	4.0
To communicate outside team	3.4

Students also commented on the value of using such a system:

- “I like being able to store info online.”
- “Easy to change things/update/store/share.”
- “It’s easy for people on a team to communicate.”
- “It kept me on schedule of what to do.”

In some cases, however, students may not recognize the value of such tools and processes until several years later, when they encounter larger projects in school or in the workforce. A former student, working in industry, emailed to describe the effects of processes and tools on his career:

“Even though I (we) complained during our software engineering class about the tedious paper work that was required for our projects, it was by far one of the most accurate representations of a ‘real world’ work environment. The processes that we underwent in your class to develop a project (especially scoping it out) has given me a leg up on my peers. I have come to understand that the (good) developers spend roughly 65% of their time planning, 20% testing and 15% coding. So thank you.”

## Lessons Learned & Recommendations

First and most importantly, remember we observed that “the major problems of our work are not so much technological as sociological in nature” (DeMarco & Lister 1999, 4, original emphasis). In other words, the biggest challenges tend to involve personalities and relationships between people; thus, teams (and teachers) need to focus on developing cohesion. Tools can help a good team to function more effectively, but can’t prevent a poor team from having problems, although in some cases tools can help diagnose problems.

Keep things simple, at least initially. Start with one or possibly two simple tools that will make the biggest difference in the project or course. Develop experience and confidence, and then decide whether to add more tools, or switch to an integrated tool. Use integrated tools, or preconfigured “best of breed” sets, rather than trying to configure multiple tools to work smoothly together. Look for open source projects that have a history of releasing new versions at least once a year, and have multiple developers and healthy user communities.

Explore tools and techniques gradually. Avoid using more than one or two new tools in a course; there are too many possibilities to explore and things that can go wrong. Some students are intimidated by new tools or lack the confidence to explore them; other students may assume that the tools will be easy, and fail to invest the effort to master them. Similarly, introduce tools and techniques to students gradually. I try to have one activity or assignment designed to give students a short, successful experience with a new tool or technique, before a larger assignment or project in which the tool or technique is essential. I usually start with communication tools, which are the most familiar to students, and then introduce collaboration and coordination tools “just in time,” depending on the course and project.

Finally, don’t give up too quickly, and be ready for occasional failures. In some cases, failure can provide an opportunity for classroom discussion about the role of tools, and the importance of matching tools to specific situations. If every experiment is a success, then we’re probably not doing enough experiments.

## Conclusions

Using open source tools in team projects benefits students in several ways. First, they will be better able to attempt and complete ambitious projects. Second, they will be exposed to tools, or types of tools, that they are likely to encounter in the future, so they will be better prepared for future courses or employment. Although learning new tools requires time and effort that could be spent on other activities, students also benefit by improving their ability to learn new tools. In some cases, however, students may not immediately see the value of tools.

Using open source tools also benefits faculty. To the extent that faculty try to manage student projects, the tools can make it easier to manage student teams and adapt the projects and teams in response to other factors. The tools can also provide insight into project and team dynamics, which can help faculty diagnose and correct problems and assess student performance more accurately. A significant challenge for faculty is keeping current with the rapidly changing array of available tools, but faculty must also become better able to learn and evaluate new tools.

## References

- Angeles, M. 2004. Using a wiki for documentation and collaborative authoring. LLRX.com, November 28. <http://www.llrx.com/features/librarywikis.htm> (accessed June 27, 2007)
- DeMarco, T., and T. Lister. 1999. *Peopleware: Productive Projects & Teams*. Dorset House.
- Bouillon, P., and J. Krinke. 2004. A platform for teaching distributed software engineering. *Proceedings of the Workshop on Cooperative Support for Distributed Software Engineering Processes*.
- Clifton, C., L.C. Kaczmarczyk, and M. Mrozek. 2007. Subverting the fundamentals sequence: Using version control to enhance course management. *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education*.

- CosmoCode. WikiMatrix - Compare them all. <http://www.wikimatrix.org> (accessed Oct 5, 2007).
- Fincher, S., M. Petre, and M. Clark. 2001. *Computer Science Project Work: Principles and Pragmatics*. Springer.
- Fountain, R. 2007. *Wiki Pedagogy*. [http://www.profetic.org/dossiers/dossier\\_imprimer.php3?id\\_rubrique=110](http://www.profetic.org/dossiers/dossier_imprimer.php3?id_rubrique=110) (accessed June 27, 2007).
- Glassy, L. 2006. Using version control to observe student software development processes. *Journal of Computing Sciences in Colleges* 21(3): 99-106.
- Hartness, K., T. N. 2006. Eclipse and CVS for group projects. *Journal of Computing Sciences in Colleges* 21(4): 217-222.
- Jones, T. 2000. *Software Assessments, Benchmarks, and Best Practices*. Addison Wesley.
- Kay, J., N. Maisonneuve, K. Yacef, and P. Reimann. 2006. The big five and visualizations of team work activity. *Proceedings of the 8th International Conference on Intelligent Tutoring Systems*.
- Kay, J., N. Maisonneuve, K. Yacef, and O. Zaiane. 2006. Mining patterns of events in students' teamwork data. *Proceedings of the Workshop on Educational Data Mining at the 8th International Conference on Intelligent Tutoring Systems*.
- Korfiatis, N., and A. Naeve. 2005. Evaluating wiki contributions using social networks: A case study on wikipedia. *Proceedings of the First On-Line Conference on Metadata and Semantics Research*.
- Kussmaul, C., S. Howe, and S. Priest. 2006. Using wikis to foster team communication, cohesion, & collaboration. *Proceedings of the 2006 American Society for Engineering Education (ASEE) Annual Conference & Exposition*.
- Kussmaul, C., and S. Albert. 2007. Reading, writing, and revising with wiki technology: Tutorial presentation. *Journal of Computing Sciences in Colleges* 22(6): 138-139.
- Lamb, B. 2004. Wide open spaces: Wikis, ready or not. *EDUCAUSE Review* 39(5): 36-48.
- Leuf, B., and W. Cunningham. 2001. *The Wiki Way: Quick Collaboration on the Web*. Addison-Wesley Professional.
- Liu, C. 2005. Using issue tracking tools to facilitate student learning of communication skills in software engineering courses. *Proceedings of the 18th Conference on Software Engineering Education & Training*.
- Liu, Y., E. Stroulia, K. Wong, and D. German. 2004. Using CVS historical information to understand how students develop software. *Proceedings of the 2004 International Workshop on Mining Software Repositories*.
- Louridas, P. 2006. Using Wikis in Software Development. *IEEE Software* 23(2): 88-91.
- MediaWiki - MediaWiki. <http://www.mediawiki.org/wiki/MediaWiki> (accessed Oct 5, 2007).
- Mierle, K., K. Laven, S. Roweis, and G. Wilson. 2005. Mining student CVS repositories for performance indicators. *Proceedings of the 2005 International Workshop on Mining Software Repositories*.
- MoinMoinWiki - MoinMoin. <http://moinmoin.wikiwikiweb.de> (accessed Oct 5, 2007).
- Moodle - A Free, Open Source Course Management System for Online Learning. <http://moodle.org> (accessed Oct 5, 2007).

- Notari, M. 2006. How to use a wiki in education: 'Wiki based effective constructive learning'. *Proceedings of the 2006 International Symposium on Wikis*.
- PmWiki. <http://pmwiki.org> (accessed Oct 5, 2007).
- Read, B. 2005. Romantic poetry meets 21st-century technology. *The Chronicle of Higher Education*.
- Reid, K.L., and G.V. Wilson. 2005. Learning by doing: Introducing version control as a way to manage student assignments. *Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education*.
- Schwartz, L., S. Clark, M. Cossarin, and J. Rudolph. 2004. Educational wikis: Features and selection criteria. *The International Review of Research in Open and Distance Learning* 5(1).
- Segetech Ltd. 2007. Bugzilla/SVN/Wiki Integration. <http://oss.segetech.com/bugzilla-svn-wiki.html> (accessed Oct 5, 2007)
- SourceForge. <http://sourceforge.net> (accessed Oct 5, 2007).
- TiddlyWiki - a reusable non-linear personal web notebook. <http://www.tiddlywiki.com/> (accessed Oct 5, 2007).
- The Trac Project - Trac. <http://trac.edgewall.org> (accessed Oct 5, 2007).
- Tomayko, J.E. 1987. *Teaching a Project-Intensive Introduction to Software Engineering*. Pittsburgh, PA: Carnegie Mellon University, Software Engineering Institute.
- Tomayko, J.E. 1998. Forging a discipline: An outline history of software engineering education. *Annals of Software Engineering* 6(1): 3-18.
- TWiki - Enterprise Wiki & Collaboration Platform. <http://twiki.org> (accessed Oct 5, 2007).
- Viégas, F.B., M. Wattenberg, and M.M. McKeon. 2007. The hidden order of Wikipedia. *Proceedings of the 12th International Conference on Human-Computer Interaction*.
- Wei, C., B. Maust, J. Barrick, E. Cuddihy, and J.H. Spyridakis. 2005. Wikis for supporting distributed collaborative writing. *Proceedings of the Society for Technical Communication 52nd Annual Conference*.
- WikiEngineComparison - MoinMoin. <http://moinmoin.wikiwikiweb.de/WikiEngineComparison> (accessed Oct 5, 2007).
- Wikipedia. <http://www.wikipedia.org> (accessed Oct 5, 2007).
- Wodehouse, A., H. Grierson, W. J. Ion, A. Juster, A. Lynn, A. L. Stone, et al. 2004. TikiWiki: A tool to support engineering design students in concept generation. *Proceedings of the International Engineering and Product Design Education Conference*.