Seppälä, Otto; Auvinen, Tapio; Karavirta, Ville; Vihavainen, Arto; Ihantola, Petri

What communication tools students use in software projects and how do different tools suit different parts of project work?

Published in:
Proceedings of the 38th International Conference on Software Engineering Companion

DOI:
10.1145/2889160.2889196

Published: 14/05/2016

Please cite the original version:
What communication tools students use in software projects and how do different tools suit different parts of project work?

Otto Seppälä
Aalto University
Dept. of Computer Science
Espoo, Finland
otto.seppala@aalto.fi

Tapio Auvinen
Aalto University
Dept. of Computer Science
Espoo, Finland
tapio.auvinen@aalto.fi

Arto Vihavainen
University of Helsinki
Dept. of Computer Science
Helsinki, Finland
avihavai@cs.helsinki.fi

Ville Karavirta
Aalto University
ville@villedkaravirta.com

Petri Ihantola
Tampere Univ. of Technology
Dept. of Pervasive Computing
Tampere, Finland
petri.ihantola@tut.fi

ABSTRACT

In software engineering education, the goal is often to provide students with authentic assignments using actual tools of the trade. Students are often allowed to select their preferred tools without specifying what to use for e.g. communication within the team, scheduling, bug tracking, etc. However, there is an abundance of tools to choose from with more appearing rapidly, which can make it difficult for students or staff to select appropriate tools for each task. In this paper, we study the suitability of several commonly used communication and collaboration tools on different tasks that students encounter in team assignments. We surveyed students’ experiences with different tools in a university-level web software development course.

1. INTRODUCTION

In software engineering education, professional skills are often practiced in projects that students perform together. These projects are typically either part of a course, or a separate project course such as a capstone project, where students get to apply their knowledge and skills in a real-world setting [2, 4]. Regardless of the form of the project, communication is at the heart of collaboration [6].

When working on a (software) project, students communicate on multiple levels. Much of the daily work is performed on the source code level, where students design and implement the functionality of specific requirements. When communicating about the work with others, ideas and designs are often shared using different modeling techniques such as UML as well as written text. On the other hand, when planning future work, discussions often evolve around (customer) requirements and features that need to be implemented to satisfy the requirements.

Today, students have an abundance of tools and approaches that they can choose from when collaborating with others. They are adept at finding relevant information, and can use a variety of communication tools to support their learning needs. As collaboration and communication tools differ in terms of their functionality, usability, target audience etc., it would be interesting to understand what are the pros and cons of commonly used tools. An understanding of the suitability of different tools for different purposes would allow us to provide students with the most suitable tools for each learning scenario. For example, if the learning goal of a team assignment is to elicit discussion of programming problems between students, what communication tools should we encourage our students to use? Up to date information on this topic is important as “new collaboration tools and associated best practices are emerging almost daily” [7].

In this study, we seek answers to the following research questions: What communication tools students use in software projects when communicating with others and how do different tools suit different parts of project work? To answer these questions, we analyzed data from 50 team projects including a total of 150 students in a university-level web software development course. The data came from two sources: 1) student interviews during project demonstrations, and 2) course feedback survey which was designed based on the interviews and sent to all students after the course. In this analysis, we focus on communication between students while, for example, use of the same tools (e.g. social media, or wikis) for searching information from the internet is out of our scope.

The rest of this paper is organized as follows. Section 2 provides details on how the research was conducted. The results of this study are introduced in Section 3, and discussion of the results in the light of the related research is provided in Section 4. Finally, Section 5 concludes the paper.
2. RESEARCH METHODS

2.1 Web Software Development Course

The study was conducted in the Web Software Development course (CSE-C3210) at Aalto university during Fall 2013–Spring 2014 (the course spans two semesters). This 5 ECTS\(^1\) course is offered in both the bachelor’s and master’s level curricula.

The primary function of the Web Software Development course is to act as a way for the students to learn and practice skills related to web development. Teamwork per se is not a learning goal on this course, but is used to provide the students a way to experience working on larger web-related projects without being overburdened by the workload that conducting such a project alone would entail. The learning goals of the course are stated as follows:

Student understands the distributed nature of a web application, session management and how the different parts of the application can communicate with each other. Student is able to design, implement, and deploy a small web application using a modern web application framework. Student knows how to test and debug a web application and has basic understanding of technologies and issues such as object relational mapping, security, efficiency, and scalability.\(^2\)

The course has three mandatory parts that all must be completed in order to pass the course – individual exercises (first half of the course), an individual exam (in the middle of the course), and a project work in groups of three students (second half of the course). Both the exercises and the exam contribute to 20% to the final grade, while the remaining 60% comes from the project. The first half of the course has been designed to build the students’ knowledge and routine that is needed in the project, and to balance out possible skill gaps between the students.

In the project work, students created a web application with a Django backend and a corresponding frontend. The teams were self-selected. This year, the topic was to create a web photo album where a user can add and edit images, captions and layouts; order and pay albums through a third party payment service, and various optional features (e.g. third party authentication and Flickr\(^3\) integration). All teams were requested to write and submit a project plan for review before starting to implement the software. To ensure sufficient time for development, the deadline for submitting the project plan was almost in the beginning of the project. At the end of the project, the teams had to have a production ready system running on the Heroku\(^4\) cloud platform, and the applications were demonstrated to teachers who then graded the projects. The project had no other intermediate deliverables or deadlines.

For version control in the project, all groups were instructed to use a private GitHub\(^5\) repository accessible also to the course staff. Apart from Django, GitHub and Heroku,

the students were free to select communication and development tools of their own liking.

2.2 Participants

About 180 students registered to the course, 150 started the project and 125 passed it. The students who attended the course had diverse backgrounds. Their programming background ranged from seasoned programmers accustomed to teamwork to novices with only little software engineering and programming experience. Out of the respondents, 13% were freshmen, 30% 2nd or 3rd year students, 35% 4th or 5th year students and the remaining 20% were at least 6th year students. As illustrated in Table 1, most students felt that they had programming experience before the project, although their web software development programming experience varied more. Teamworking skills were also scattered.

2.3 Interviews and Survey

After the course, all students were requested to answer an online survey regarding the course and students’ backgrounds. While the students were encouraged to answer the survey, answering was not mandatory, and the students were not rewarded in any way for doing so. Finally, a total of 34% (n=45) of the students answered the survey.

To find out what communication or collaboration tools students use and what are the tools’ pros and cons, we asked the following questions in our survey: 1) what tools did you use, and 2) what did you use them for. Survey questions were formulated based on the interviews during the project demo sessions. As collaboration tools were informally mentioned by almost all groups, we were able to provide a predefined list of tools from where students selected the ones they had used. We focused on collaboration and communication tools that are supposed to be used by more than one team member and excluded tools meant for individual use. For each communication medium, we asked what was the value of that tool in various tasks: brainstorming, scheduling meetings, scheduling deadlines, distributing tasks, solving programming problems, and keeping up the team spirit. Students did not rate tools that they did not report using.

3. RESULTS

The communication methods identified during the interviews included face-to-face communication, IRC (Internet Relay Chat), email, video or voice conferences (e.g. Skype), social media (i.e. Facebook), Flowdock\(^6\), Trello\(^7\), GitHub\(^8\) Issues, private wikis (e.g. GitHub Wiki), WhatsApp, Piazza\(^9\), Skype chat (no audio), and Google’s chat services. Table 2 summarizes students’ judgments of the suitability of different methods for all the tasks included in the survey.

There are a few features to be noted. Brainstorming is a highly interactive activity and the results here are not surprising. Face to face communication reigns over all the alternatives, with all the tools that offer some form of interactive chat coming second. Solving programming problems has a very similar preference pattern to brainstorming for the generic chatting tools but here the tools dedicated to sup-

\(^1\)European Credit Transfer System, 1 ECTS corresponds to roughly 27 hours of work.

\(^2\)https://education.github.com/

\(^3\)https://trello.com/

\(^4\)https://github.com/

\(^5\)https://piazza.com/
Table 1: Students’ self-reported prior programming and teamwork experience. 1=Strongly Disagree ... 4=Strongly Agree

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>I had programming experience</td>
<td>-</td>
<td>1 (3%)</td>
<td>7 (18%)</td>
<td>30 (79%)</td>
</tr>
<tr>
<td>I had web software development experience</td>
<td>7 (18%)</td>
<td>11 (29%)</td>
<td>10 (26%)</td>
<td>10 (26%)</td>
</tr>
<tr>
<td>I had experience from teamwork in software projects</td>
<td>3 (8%)</td>
<td>8 (21%)</td>
<td>9 (24%)</td>
<td>18 (47%)</td>
</tr>
</tbody>
</table>

Table 2: Students’ judgements of the suitability of different communication methods for various tasks. Means where there are 2 or less respondents are in parenthesis. 1=Strongly disagree ... 4=Strongly agree.

<table>
<thead>
<tr>
<th></th>
<th>brain storming</th>
<th>programming problems</th>
<th>task distribution</th>
<th>scheduling deadlines</th>
<th>scheduling meetings</th>
<th>team spirit</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>face-to-face</td>
<td>3.4</td>
<td>2.9</td>
<td>3.0</td>
<td>2.2</td>
<td>2.0</td>
<td>3.0</td>
<td>31</td>
</tr>
<tr>
<td>IRC</td>
<td>1.9</td>
<td>2.2</td>
<td>2.5</td>
<td>3.3</td>
<td>3.6</td>
<td>2.4</td>
<td>12</td>
</tr>
<tr>
<td>email</td>
<td>1.7</td>
<td>1.9</td>
<td>2.7</td>
<td>2.8</td>
<td>2.0</td>
<td>1.3</td>
<td>19</td>
</tr>
<tr>
<td>GitHub issues</td>
<td>1.8</td>
<td>3.2</td>
<td>2.8</td>
<td>2.0</td>
<td>1.3</td>
<td>1.9</td>
<td>19</td>
</tr>
<tr>
<td>Facebook</td>
<td>2.6</td>
<td>2.7</td>
<td>2.8</td>
<td>3.0</td>
<td>3.5</td>
<td>2.8</td>
<td>7</td>
</tr>
<tr>
<td>Flowdock</td>
<td>2.8</td>
<td>3.5</td>
<td>3.5</td>
<td>3.7</td>
<td>3.8</td>
<td>3.2</td>
<td>6</td>
</tr>
<tr>
<td>Trello</td>
<td>1.4</td>
<td>1.3</td>
<td>3.8</td>
<td>2.5</td>
<td>1.0</td>
<td>2.3</td>
<td>6</td>
</tr>
<tr>
<td>video chat</td>
<td>3.0</td>
<td>2.5</td>
<td>2.6</td>
<td>2.5</td>
<td>(2.5)</td>
<td>2.8</td>
<td>5</td>
</tr>
<tr>
<td>Skype chat(text)</td>
<td>3.0</td>
<td>3.7</td>
<td>(3.0)</td>
<td>(4.0)</td>
<td>(2.5)</td>
<td>(4.0)</td>
<td>4</td>
</tr>
<tr>
<td>any wiki</td>
<td>(2.0)</td>
<td>(3.0)</td>
<td>(1.0)</td>
<td>(1.0)</td>
<td>(1.0)</td>
<td>(1.0)</td>
<td>2</td>
</tr>
<tr>
<td>Google chat</td>
<td>(3.0)</td>
<td>(3.0)</td>
<td>(2.0)</td>
<td>(1.0)</td>
<td>(2.0)</td>
<td>(3.0)</td>
<td>1</td>
</tr>
</tbody>
</table>

Scheduling face-to-face meetings seems to be the task that requires least specialization from the tool, all the tools providing chat or email being viable options. The generic tools are a little less suitable for scheduling deadlines though, Flowdock getting the best score. For the last activity column, team spirit, there is no clear winner. Of the lesser used tools, some students reported collaborating using Skype to share a view of a desktop.

4. DISCUSSION

We found students to use a versatile set of communication and collaboration tools, including tools specifically designed for teamwork and communication in software projects. The list of tools is arguably more specialized than what is reported in previous research, e.g., by Clear et al. [2] who list a few options to facilitate the discussion between students and staff in capstone projects.

It can be observed in Table 2 that most collaboration tools work equally well in tasks that are less technically oriented (e.g., brainstorming, agreeing on meeting times). On the other hand, tasks such as agreeing on project deadlines, task division, and solving programming problems are tasks where students seem to prefer dedicated tools. That is, dedicated tools typically beat generic tools in more technically oriented tasks. In addition, teams that used dedicated tools were clearly satisfied with their choices.

For comparison, Paasivaara et al. [9] conducted a study where they “asked the students to evaluate the usefulness of different tools they used for creating a shared understanding in the project, for developing a sense of teamwork, and for completing the project”. They observed only minor differences between Google Hangout (video chat), Google docs, email, git, and Agileant for the previously listed goals. A tool called Jazz was consistently perceived less useful in supporting the goals. The difference between Paasivaara et al. and our results could be explained, in our opinion, by the fact that the tools and goals listed by Paasivaara et al. were more generic. Another explanation could be that the teams studied by Paasivaara et al. were distributed whereas our teams had the possibility to meet face-to-face.

As the students chose the tools by themselves and adjusted their working behaviors accordingly, our results are in line with the work by Conole et al. [3] who suggest that students are independent and able to identify and adapt the tools that they need for learning. From the teacher’s perspective, these tools that were identified in the student interviews can be offered as a baseline for students who look to adapt a tool or few into their working practices.

Overall, the results make it clear that communication tools are important for student software developers. Students ranked face-to-face communication as the most preferred communication method only for brainstorming, while software tools scored higher in other tasks. Our hypothesis is that this effect can be partially explained by the level of response and immediateness of the task; while students benefit from instant feedback during brainstorming, such immediateness is not needed in e.g. task distribution, where students value tools dedicated for software development.

Although some tools are in some cases perceived more suitable than face-to-face communication, it should be noted that face-to-face was still the most common communication
method in our study. It is natural and one of the preferred communication ways in e.g. Agile methodologies [1] which emphasizes the power of face-to-face communication in conveying understanding. However, as face-to-face communication leaves no history except for a memory trace, one should pay attention to making notes. Different communication tools also have different approaches to storing past communication, making some tools more suitable for certain tasks.

Our result of face-to-face communication being the most popular approach in brainstorming, closely followed by video conferencing and chat and email is inline with the findings of Niinimäki et al. [8] who argue that these activities are the best for uncertain and ambiguous tasks aiming at building common understanding. Moreover, Niinimäki et al. also argue that for simple and concise tasks aiming at sharing information, email is a good option. Our findings also support this as email, although not the best in any task, was at its best when scheduling meetings or deadlines.

Finally, the spectrum of cooperative work and collaboration tools is often characterized by using Johansen’s time-space matrix [5]. As illustrated in Table 3, the model makes a distinction between collaboration where users are in the same place and where they are separated by distance, as well as activities where participants can act simultaneously and where they act asynchronously. Interestingly, we failed to identify asynchronous but co-located communication methods (e.g., teams own whiteboard they could use throughout the project) in our study. For further research, it would be interesting to see how students would use such facilities in software projects.

Table 3: Johansen’s time-space matrix with example tools and media. A single tool can be classified into are multiple corners of the matrix.

<table>
<thead>
<tr>
<th>same place</th>
<th>different place</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>same time</strong></td>
<td><strong>different time</strong></td>
</tr>
<tr>
<td>face-to-face</td>
<td>e.g., team rooms</td>
</tr>
<tr>
<td>video conference</td>
<td>email, wiki, Github issues, Facebook, Flowlodc, IRC</td>
</tr>
<tr>
<td>chat apps., Facebook, Flowdoc, IRC</td>
<td>Facebook, Trello, IRC (*nix screen)</td>
</tr>
</tbody>
</table>

4.1 Limitations
As is typical for studies in educational settings, it is possible that our results depend on the context and culture. For example, in Finland it is quite typical that students enter working life as e.g. junior software developers already during their studies. Thus, it is possible that the programming experience of the course participants is higher than it would be in other contexts; naturally the chosen pedagogy and teaching practices in earlier courses play a role here as well, as do other courses that the students are enrolled in.

The survey in this study was conducted as a post-course survey. Thus, the responses are from students who were asked to recall their own feelings and preferences during the project, parts of which may have already been forgotten. Furthermore, we had low response rates in some of the questions which may undermine the reliability of the results. Moreover, as only a subset of students answered the survey (34% of students that finished the course), it is possible that an inherent selection bias exists. In addition, the survey was formulated so that we asked only about the tools student had really used. It is possible that a student tried a tool, then abandoned it for any reason, and finally did not express his or her opinions on the tool that was not really used. Thus, everyone who described a tool in our survey liked the tool at least enough to stick with it. To overcome these limitations in the future, we will look into asking students to report the tools used during the course instead of a post-course survey.

5. CONCLUSIONS
Communication tools and services aimed for software development are rapidly evolving. They do not only offer messaging, version controlling, and issue management any more. There are various dedicated tools targeted for improving communication in software projects and at least some students like to be their early adopters. Indeed, brainstorming seems to be the only activity where students found face-to-face communication more effective than using dedicated tools. We encourage teachers to follow the development of this field and start by looking at what kinds of tools our students found beneficial as discussed in Section 3.

Acknowledgements
The work is partially funded by Tekes/Digile Need4Speed program.

6. REFERENCES