Exploring Pair Programming in Distributed Object-Oriented Team Projects

Prashant Baheti  
Department of Computer Science  
North Carolina State University  
Raleigh, NC 27695  
+1 919-755-1264  
ppbaheti@unity.ncsu.edu

Dr Laurie Williams  
Department of Computer Science  
North Carolina State University  
Raleigh, NC 27695  
+1 919-515-7925  
williams@csc.ncsu.edu

Dr Edward Gehringer  
Department of Computer Science  
Dept. of ECE  
North Carolina State University  
Raleigh, NC 27695  
+1 919-515-2066  
efg@ncsu.edu

Dr David Stotts  
Department of Computer Science  
University of North Carolina at Chapel Hill  
Chapel Hill, NC 27599  
+1-919-962-1833  
stotts@cs.unc.edu

ABSTRACT

Previous research [1, 4] has indicated that pair programming is better than individual programming when the pairs are physically collocated. However, important questions arise: How effective is pair programming if the pairs are not physically next to each other? What if the programmers are geographically distributed? An experiment was conducted at North Carolina State University to compare the different working arrangements of student teams developing object-oriented software. The teams were both collocated and in distributed environments; some teams practiced pair programming while others did not. The results of the experiment indicate that it is feasible to develop software using distributed pair programming, and that the resulting software is comparable to software developed in colocated or virtual teams. Our findings will be of significant help to educators dealing with team projects for distance-learning students, as well as organizations that are involved in distributed development of software.

Keywords

Extreme Programming (XP), collocated, distributed, pair programming, collaborative programming, distance education, virtual team.

1. INTRODUCTION

Distance education has come into prominence in recent years. Team projects in DE computer-science courses call for distributed development. These teams need to communicate and work effectively and productively. Through the vehicle of groupware, team members can communicate with each other and complete their projects even when they are remotely located, or when they work at incompatible hours. Distributed team projects are also very common in the software industry. Employing the power of distributed development can increase an organization's opportunities to win new work by opening up a broader skill and product knowledge base, coupled with a deeper pool of personnel to potentially employ [3]. Major corporations have launched global teams, expecting that technology will make “virtual collocation” a feasible alternative [2].

Previous research [1, 4] has indicated that pair programming is better than individual programming in a collocated environment. Do these results also apply to distributed pairs? It has been established that distance matters [2]; face-to-face pair programmers will most likely outperform distributed pair programmers in terms of sheer productivity. However, the inevitability of distributed work in DE calls for research in determining how to make this type of work most effective. This is the focus of this paper. We believe that our results also have implications for industry, where virtual teams are quite common.

The rest of the paper is organized as follows. Section 2 describes the previous work done with respect to pair programming and virtual teams. Section 3 gives the hypothesis for which we test our results. Section 4 outlines the experiment that was conducted in a graduate class at NC State University. Section 5 presents the results. It is followed by an outline of future work in Section 6. The conclusions are presented in Section 7.

2. PREVIOUS WORK

2.1 Pair Programming

Pair programming is a style of programming in which two programmers work side by side at one computer, continuously collaborating on the same design, algorithm, code or test. One of the pair, called the driver, is typing at the computer or writing down a design. The other partner, called the navigator, has many jobs. One of the roles of the navigator is to observe the work of the driver, looking for tactical and strategic defects in the work of the driver. Tactical defects are syntax errors, typos, calls to the wrong method, etc. Strategic defects are said to occur when the team is headed down the wrong path — what they are
implementing won’t accomplish what it needs to accomplish. Any of us can be guilty of straying off the path. A simple, “Can you explain what you’re doing?” from the navigator can serve to bring the driver back onto the right track. The navigator has a much more objective point of view and can better think strategically about the direction of the work. The driver and navigator can brainstorm on demand at any time. An effective pair-programming relationship is very active. The driver and the navigator communicate at least every 45 seconds to a minute. It is also very important for them to switch roles periodically. Note that pair programming includes all phases of the development process — design, debugging, testing, etc. — not just coding. Experience shows that programmers can pair at any time during development, in particular when they are working on something that is complex. The more complex the task, the greater the need for two brains [1, 9].

Research has shown that pairs finish in about half the time of individuals and produce higher quality code. The technique has also been shown to assist programmers in enhancing their technical skills, to improve team communication, and to be more enjoyable [1, 9, 10, 11].

2.2 Virtual Teaming

A virtual team can be defined as a group of people, who work together towards a common goal, but across time, distance, culture and organizational boundaries [15]. In our context the goal is development of software. The members of a virtual team may be located at different work sites, or they may travel frequently, and need to rely upon communication technologies to share information, collaborate, and coordinate their work efforts. As the business environment becomes more global and businesses are increasingly in search of more creative ways to reduce operating costs, the concept of virtual teams is of paramount importance [6].

Distance education may be defined as “a form of education in which there is normally a separation between teacher and learner, and thus one in which other means — the printed and written word, the telephone, computer conferencing or teleconferencing, for example — are used to bridge the physical gap” [14]. The concept of virtual teaming is a boon for distance education as it allows distance-learning students participate in team projects, although the individual team members are geographically dispersed.

In the past, there was no support for collaborative programming for virtual teams. Advancements in technology and the invention of groupware have changed this situation. “Students can now work collaboratively and interact with each other and with their teacher on a regular basis. Students develop interpersonal and communication skills that were unavailable when working in isolation” [16].

Distributed learning, or distance education, is experiencing explosive growth. “Online learning is already a $2 billion business; Gerald Odening, an analyst with Chase Bank, predicts that the figure will rise by 35% a year, reaching $9 billion by 2005” [12]. The federal government assigns great importance to advances in distributed learning. In November 1997, the Department of Defense (DoD) and the White House Office of Science and Technology Policy (OSTP) launched the Advanced Distributed Learning (ADL) initiative. The role of the ADL is “to ensure access to high-quality education and training materials that can be tailored to individual learner needs and made available whenever and wherever they are required” [13]. Programming students have been major participants in the growth of distance education. However, software project courses, particularly team projects, provide a significant challenge to geographically separated students.

A primary consideration in virtual teaming is that of communication [7]. Poor communication can cause problems like inadequate project visibility, wherein everyone does his/her individual work, but no one knows if the pieces can be integrated into a complete solution. Coordination among the team members could also be a problem. Finally, the technology used must be robust enough to support distributed development.

In the educational field, with distance learning on the rise, virtual teaming has become inevitable. At the same time, it is important to meet the same learning objectives in distance learning as in a traditional classroom.

3. HYPOTHESES

In the fall of 2001, we ran an initial experiment at North Carolina State University to assess whether geographically distributed programmers benefit from using technology to collaborate synchronously with each other. Specifically, we examined the following hypotheses:

- Distributed teams whose members pair synchronously with each other will produce higher quality code than distributed teams that do not pair synchronously.
- Distributed teams whose members pair synchronously will be more productive (in terms of LOC/hr.) than distributed teams that do not pair synchronously.
- Distributed teams who pair synchronously will have comparable productivity and quality when compared with collocated teams.
- Distributed teams who pair synchronously will have better communication and teamwork within the team when compared with distributed teams that do not pair synchronously.

4. THE EXPERIMENT

The experiment was conducted in a graduate class, Object-Oriented Languages and Systems, taught by Dr Edward Gehring at North Carolina State University. The course introduces students to object technology and covers

---

1. [http://courses.ncsu.edu/csc517/common](http://courses.ncsu.edu/csc517/common)
The VBEE program is moving from videotape to video servers, but this change is not yet complete.
distributed pair programming is superior to collocated pair programming for student teams. Our goal was to demonstrate that distributed pairing is a viable and desirable alternative for use with student teams, particularly for distance education students. We plan to repeat this experiment in the Fall 2002 semester to build up a larger base of results.

5.1 Productivity
Productivity was measured in terms of lines of code per hour. Average lines of code per hour for the four environments are shown in Figure 1.

![Box plot for LOC/hr](image)

The results show that distributed teams had a slightly greater productivity as compared to collocated teams; however, the f-test for the four categories shows that results are not statistically significant ($p < 0.1$), due to high variance in the data for distributed groups. This is better depicted by the box plot (Figure 2) for the four categories, which illustrates the distribution of the metric for the four environments.

shows the range in which the bottom 25% of the values lie. Thus, the end points of the two line segments indicate the total range that the values for a particular category fall into. For example, the median for the non-pair collocated category is around 10 LOC/hr., with the middle 50% of the values lying between approximately 9 and 13 LOC/hr., while the entire range is between 5 and 35 LOC/hr., approximately.

If the comparison is restricted to the two distributed categories, a statistical t-test on the two categories shows that this difference is not statistically significant. In terms of productivity, the groups involved in virtual teaming (without pairs) is not statistically significantly better than those involved in distributed pair programming. In other words, teams involved in distributed pair programming are not shown to be worse in terms of productivity than those forming virtual teams without distributed pair programming.

5.2 Quality
The quality of the software developed by the groups was measured in terms of the average grade obtained by the group out of a maximum of 110. The graph below indicates that the performance of students did not vary much from one category to another.

![Grades](image)

A box plot for the grades only corroborates the claim made above. Although nothing statistically significant can be said about the grades for the four categories, it is interesting to see that those teams performing distributed pair programming were very successful in comparison to other groups. The results of the statistical tests indicate that teams involved in virtual teaming were not significantly better than the distributed teams using pair programming, in terms of grade.
5.3 Student Feedback

Productivity and product quality is important. However, as educators we strive to provide positive learning experiences for our students. We ran a survey to assess students’ satisfaction with their working arrangement. One of the questions was about cooperation within the team. Table 1 shows the responses of the students in the different environments.

The communication among the team members is an important issue in team projects. Table 2 shows the responses of students regarding communication among team members.

<table>
<thead>
<tr>
<th></th>
<th>Very good</th>
<th>Good</th>
<th>Fair</th>
<th>Poor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-pair collocated</td>
<td>46%</td>
<td>40%</td>
<td>11%</td>
<td>3%</td>
</tr>
<tr>
<td>Pair collocated</td>
<td>62%</td>
<td>28%</td>
<td>10%</td>
<td>0%</td>
</tr>
<tr>
<td>Non-pair distributed</td>
<td>45%</td>
<td>37%</td>
<td>18%</td>
<td>0%</td>
</tr>
<tr>
<td>Pair distributed</td>
<td>83%</td>
<td>17%</td>
<td>0%</td>
<td>0%</td>
</tr>
</tbody>
</table>

Responses to the question, “How was the cooperation between your team members?”

Table 1: Cooperation within team

<table>
<thead>
<tr>
<th></th>
<th>Very good</th>
<th>Good</th>
<th>Fair</th>
<th>Poor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-pair collocated</td>
<td>57%</td>
<td>26%</td>
<td>11%</td>
<td>6%</td>
</tr>
<tr>
<td>Pair collocated</td>
<td>58%</td>
<td>28%</td>
<td>12%</td>
<td>2%</td>
</tr>
<tr>
<td>Non-pair distributed</td>
<td>41%</td>
<td>41%</td>
<td>14%</td>
<td>4%</td>
</tr>
<tr>
<td>Pair distributed</td>
<td>67%</td>
<td>33%</td>
<td>0%</td>
<td>0%</td>
</tr>
</tbody>
</table>

Responses to the question, “How was the communication with your team?”

Table 2: Communication among Team Members

The survey also indicates that five out of six students felt that coding and testing are most suitable phases for distributed pair programming. Collocated pair programmers, in general, found pair programming to be useful in all the phases of software development. When asked to identify the greatest obstacle to distributed pair programming, students commented as follows:

“Initially exchanging code/docs via e-mail was a problem. Later on we used Yahoo briefcases to upload code to others to read it from there. From then on things went very smooth”

“Finding common time available for all.”

The students were asked to identify the biggest benefits of the distributed pair programming, and responded—

“If each person understands their role and fulfills their commitment, completing the project becomes a piece of cake. It is like Extreme Programming with no hassles. If we do not know one area we can quickly consult others in the team. It was great.”

“There is more than one brain to work on the problem.”

“It makes the distance between two people very short.”

Five out of the six students involved in distributed pair programming thought that technology was not much of a hindrance in collaborative programming. Also, 23 out of 28 students involved in virtual teaming with or without pair programming felt that there was proper cooperation among team members.

6. FUTURE WORK

The experiment we conducted was a classroom experiment among 132 students, including 34 distance-learning students. To be able to draw statistically significant conclusions, such experiments have to be repeated, on a larger scale if possible. However, this experiment has given initial indications of the viability of distributed pair programming.

We intend to conduct more experiments like this so that we can draw conclusions about distributed pair programming, and whether virtual teams should be a standard practice in the classroom as well as in industry.

7. CONCLUSIONS

The results of our experiment indicate the following:

Pair programming in virtual teams is a feasible way of developing object-oriented software.

Pair programming in collocated teams is a feasible way of developing object-oriented software.

Software development involving distributed pair programming seems to be comparable to collocated software development in terms of two metrics, namely productivity (in terms of lines of code per hour) and quality (in terms of the grades obtained).

Collocated teams did not achieve statistically significantly better results than the distributed teams.
Feedback from the students indicates that distributed pair programming fosters teamwork and communication within a virtual team.

Thus, the experiment conducted at NC State University is a first indication that distributed pair programming is a feasible and efficient method for dealing with team projects.

8. ACKNOWLEDGMENTS
We would like to thank NCSU undergraduate student Matt Senter for his help in administering this experiment. The support of Intel in providing equipment is graciously acknowledged. We would also like to thank NCSU graduate student Vinay Ramachandran for developing the tool called Bryce to record project metrics.

9. REFERENCES