Cost Estimation & Metrics
Fall 2016

GUIDED BY
Prof. YE YANG

Submitted by
Prabhjot Singh
Team 8
Pair Programming In SDLC

(To what extent does Pair Programming Improve the process of software development)
Summary

In this report I have tried to analyze the different aspects of pair programming in terms of code quality, code productivity and employee satisfaction. I have also tried to estimate the appropriate expertise level of both the programmers to successfully implement pair programming and for which tasks pair programming is most useful. Overall I have divided my whole research into four primary goals:

1. Evaluate effectiveness of pair programming in terms of code quality.
2. Evaluate effectiveness of pair programming in terms of productivity.
3. Evaluate effectiveness of pair programming in terms of employee satisfaction.
4. Evaluate expertise level of both the programmers and nature of development task for successful implementation of pair programming.

So different metrics to evaluate these goals were determined and the goals were answered to evaluate the role of pair programming in software development life cycle. For the data collection the existing publications were studied and conclusions from them were made. Several online forums such as google forms, Quora, Stack Overflow were used to collect data from developers about their experience with pair programming. The data collected was used to evaluate the goals and the results were compared to the conclusions from the existing publications. Then the final conclusion was made to determine what impact does pair programming has on software development. In this analysis and research, I have tried to give a complete picture of pair programming concerning its effect with respect to different measures as proposed in the GQM.
# Table of Contents

<table>
<thead>
<tr>
<th>Number</th>
<th>Section</th>
<th>Page No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Summary</td>
<td>3</td>
</tr>
<tr>
<td>2.</td>
<td>Introduction</td>
<td>5</td>
</tr>
<tr>
<td>3.</td>
<td>Proposed Metrics</td>
<td>6</td>
</tr>
<tr>
<td>4.</td>
<td>Results and Discussion</td>
<td>7</td>
</tr>
<tr>
<td>5.</td>
<td>Conclusions</td>
<td>20</td>
</tr>
<tr>
<td>6.</td>
<td>Limitations</td>
<td>21</td>
</tr>
<tr>
<td>7.</td>
<td>Reflection</td>
<td>21</td>
</tr>
<tr>
<td>8.</td>
<td>References</td>
<td>22</td>
</tr>
<tr>
<td>9.</td>
<td>List of Figures</td>
<td>22</td>
</tr>
</tbody>
</table>
**Introduction**

Agile development techniques are now widely followed and pair programming is an agile software development technique in which two programmers work together at one workstation. Meaning one is typing and one is observing. There are different approaches and best practices of how to best co-operate. Most commonly the one who is typing is called the driver. The person beside is called the Navigator. The Navigator will focus more on detecting bugs and problems, structure and what to focus on next. The idea is to split up responsibility between the driver and navigator to keep a steady development flow, avoiding bugs, better structure and while doing that share more knowledge across the team. The driver and navigator should be constantly talking to keep both involved.

Pair programming is widely followed in software development and it would be important to evaluate how it helps in producing a better quality code in lesser time. Pair programming (sometimes called peer programming) is a controversial topic. Some developers love it and some grow horns of hate as soon as they hear the words. Whoever I ask or whatever articles I read there seems to be a strong opinion either for or against. As the agile techniques are very popular and I am a software engineering student I am also very curious to know about the impacts of pair programming in Software Development. To solve the dilemma about Pair Programming, I’ve done some research in an attempt to bring some clarity as how its important and how and when should it be performed.

The information gathered could be very beneficial for the developers and project managers, whether to implement the pair programming technique in development process or not. It is also very helpful for new organizations to decide as whether to implement pair programming in their organization or not. It is also very important to determine, as what should be the expertise level of the developers to use pair programming and for which tasks it is more beneficial.

**For the data collection** I have used online forums to get responses from different developers. I have interviewed my fellow developer friends including Professor James Rowland, to have the questions answered according to their prior experience with pair programming. The results from the data collected were then compared to the results from the existing publications to validate them and evaluate the required goals for the project.
Proposed Metrics:

I have proposed the following GQM for my research and the corresponding metrics and questions for it. So the data corresponding these goals were obtained according to the metrics defined and thus the goals were evaluated and the conclusions were made based on the data collected.

Goal1- Evaluate effectiveness of pair programming in terms of code quality.

Q: How successfully is the code implemented?
Metrics - test cases passed
Q: How easy is to maintain the code?
Metrics - complexity, maintainability index
Q: How many developers feels that pair programming produces better quality?
Metrics – no. of programmers

Goal2- Evaluate effectiveness of pair programming in terms of productivity.

Q: How much effort is required to successfully implement the code?
Metrics - staff months, staff days
Q: How much time is taken to deliver the code?
Metrics – calendar years

Goal3- Evaluate effectiveness of pair programming in terms of employee satisfaction.

Q: How well is pair programming received?
Metrics - no of developers using pair programming, review rating of pair programmers

Goal4- Evaluate expertise level of both the programmers and nature of the development task for successful implementation of pair programming.

Q: What is the prior experience of the developer doing pair programming?
Metrics - no of years
Q: What type of tasks should pair programming be used for?
Metrics – task type (complex task, testing task etc.)
Results and Discussion:

So now analyzing each goal with respect to its metrics based on the data collected. For the first goal:

Goal1- Evaluate effectiveness of pair programming in terms of code quality.

Q: How successfully is the code implemented?
Metrics - test cases passed
Q: How easy is to maintain the code?
Metrics - complexity, maintainability index
Q: How many developers feels that pair programming produces better quality?
Metrics – no. of programmers

- Following are the details which I obtained about Pair Programming about code quality:
- I got response from 36 developers through online forums out of which 24 developers believed that pair programming is beneficial in terms of code quality and producing lesser number of defects.
- 2 of the developers feel that it depends on the problem at hand
- 10 of the developers were against using Pair Programming

![Programmer Preference on Pair Programming based on Code Quality](image)

- prefer pair programming in terms of code quality
- prefer individual programming in terms of code quality
- undecided (depends on job)
I interviewed 9 of my developer friends for pair programming out of which 6 feel that pair programming is beneficial and leads to better results in terms of code quality. 3 of them are undecided about the effects of pair programming in terms of code quality because in their previous projects sometimes pair programming has led to better quality and sometimes it was not effective.

So analyzing the total results from the different developers through online forums and my fellow friends

30 out 45 feel that pair programming is helpful in producing a better quality code.

\( \frac{30}{45} \times 100 = 66.6\% \)

10 out of 45 developers feel that pair programming is not helpful in producing a better quality code

\( \frac{10}{45} \times 100 = 22.2\% \)
5 out of 45 developers are not sure about pair programming in terms of code quality

\((5/45) \times 100 = 11.2\%\)

Responses of developers in terms of test cases passed.

So I planned my questions in way that I gave four options to the developers in terms of test cases passed by doing pair programming.

The four options were

1. No effect on test cases
2. Greater than 10% more test cases passed
3. Greater than 25% more test cases passed
4. Greater than 50% more test cases passed
Out of 36 responses I got from online forums, 16 developers feel there are 25% more test cases passed because of pair programming, 8 programmers feel that there 15% more test cases passed because of pair programming. Rest feel that it does not lead to more test cases passed.

Now analyzing the results from existing publications and comparing them.

Taken from the publication: The Costs and Benefits of Pair Programming by Alistair Cockburn and Laurie Williams

In 1999, a controlled experiment run at the University of Utah investigated the economics of pair programming. Advanced undergraduates in a Software Engineering course participated in the experiment.
So in this experiment the test cases passed by individual programmers and pair programmers were compared and it indicates more number of test cases passed as in case of pair programming. More test cases passed means lesser bugs and thus indicating better code quality. The result from this publication indicates that there are about 15% more test cases passed in case of pair programming. The data from online forums also indicates about 15% more test cases passed if we compute the average from different responses.

\[
\frac{(25 \times 16) + (8 \times 15)}{36} = 14.45,
\]

So responses from online forums also indicates about 15 % increase in test cases passed.
In the quantitative study at the University of Utah, the pairs not only completed their programs with superior quality, but they consistently implemented the same functionality as the individuals in fewer lines of code. So this also indicates better maintainability and lesser complexity.

So here we see the results from responses of developers through interviews and online forums and compare it to the existing publications, we see that both the results do indicate that pair programming leads to better quality, lesser bugs, better maintainability and lesser complexity.

Now Analyzing the second goal

**Goal2- Evaluate effectiveness of pair programming in terms of productivity.**

Q: how much effort is required to successfully implement the code?
Metrics - staff months, staff days
Q: how much time is taken to deliver the code?
Metrics – calendar years

Results of data collected from online forums:

I had 36 responses from developers through online forums and interviewed 9 of my friends for their experience with pair programming.

24 of the developer’s responses through online forums and 6 of my friends felt that pair programming leads to lesser development time and is more productive.

So in total (24 + 6) = 30 developers feel that pair programming is more productive and 15 believe that it is not more productive

So in total

(30/45) * 100 = 66.6%

We have more percentage of developers who feel that the Pair Programming is more productive.
So now talking the views who feel that pair programming is more productive

When asked from developers how much lesser effort do pair programming leads to. These were responses.

17 feel about 20% lesser effort
11 feel about 30% lesser effort
2 feel about 40% lesser effort

When asked from developers how much lesser time do they feel that pair programming takes to deliver the project. These were responses.
12 feel about 20% lesser time
9 feet about 30% lesser time
9 feel about 40% lesser time

These were the responses they gave based on their experience with previous projects and their conclusions from that

Now analyzing the work from existing publications:

Taken from the publication: The Costs and Benefits of Pair Programming by Alistair Cockburn and Laurie Williams

In 1999, a controlled experiment run by the second author at the University of Utah investigated the economics of pair programming. Advanced undergraduates in a Software Engineering course participated in the experiment. One third of the class coded class projects as they had for years – by themselves. The rest of the class completed their projects with a collaborative partner.

The pairs only spent about 15% more time on the program than the individuals. The second figure shows the post development test cases the students passed for each program.

The total time of pair programmers may be more but when working in tandem pair programmers deliver the product faster. And as in pair programmer’s code more test cases are passed, it means lesser bugs and less time and money is spent in correcting them. Thus these factors make pair programming more productive.
Taken from the publication:

Strengthening the Case for Pair Programming

Comparison of pair programmers’ and individuals’ project completion times.

So this data is taken from an experiment in this publication to verify and compare the data reported by the developers that pair programming leads to lesser time in project completion.

So as the graph shows

By working in tandem, the pairs completed their assignments 40% to 50% faster.
Now analyzing the third goal:

**Goal3- Evaluate effectiveness of pair programming in terms of employee satisfaction.**

Q: How well is pair programming received?
metrics -no of developers using pair programming

I got response from 36 developers through online forums out of which 21 developers like to work in pairs. Out of the remaining 15, 3 feel pair programming may lead to better code but they don’t like to code sitting with an another person all the time. 12 of the developers don’t like to code in pairs. I also interviewed 9 of my developer friends for pair programming out of which 6 feel very positive about working in pairs.

So in total we have \((36 + 9) = 45\) responses. 27 developers prefer working in pairs. 18 developers prefer working alone.

So as we see in this data that developers in majority prefer working in pairs. So it can be ascertained that pair programming does have better rating in terms of employee satisfaction as more developers like to work in pairs. This result can also be compared from the results in the publication: The Costs and Benefits of Pair Programming by Alistair Cockburn and Laurie Williams.
In statistically significant results, pair programming teams who had earlier programmed alone reported that they enjoyed pair programming more. The graph shows results of anonymous surveys of professional pair programmers and of student pair programmers at the University of Utah.

Goal 4- Evaluate expertise level of both the programmers and nature of development task for successful implementation of pair programming.

Q: What is the prior experience of the developer doing pair programming?
Metrics - No of years
Q: What type of tasks should pair programming be used for?
Metrics – Task type (complex task, testing task etc.)

I got response from 45 developers out of which 24 feel that the expertise level of the programmers should be same i.e. both the programmers should have worked same no of years, to get more effective results from pair programming. Others feel that it does not matter with respect to the expertise level of the programmers.
I also asked developers as for what type of tasks should we use Pair Programming. I gave them the following options:

Program design, development, looking for errors and every programming task.

I told them to give a rating of 1 to 5 for each option, with one indicating pair programming least suitable and 5 indicating pair programming most suitable for the job. So as I have total data from 45 developers out of which 30 were in favor of Pair Programming. So analyzing the responses of 30 developers.

Total points for each option and computing the average.

1. Program design - 101, Average = (101/30) = 3.37
2. Complex programming task – 114, Average = (114/30) = 3.8
3. Looking for errors – 128, Average = 128/30 = 4.27
4. Every programming task – 64

Average = 64/30 = 2.13
So according to data above the pair programming techniques should be most used in case of error finding and developers least prefer it to use for every programming task.

The results can be compared with the results from existing publication

So this data is taken from: Pair programming in software development teams – An empirical study of its benefits

Here also an analysis is done as in which programming tasks is pair programming most helpful. Here we see significant similarity with the current responses from developers as here also the most appropriate task associated with pair programming is looking for errors.

Fig. 4.
Answers to the question “For which task is working with a partner useful?”.
Conclusions:

So now concluding from all the data collected through online forums and interviews and also by comparing it to the existing publications.

1. For the first goal to evaluate effectiveness of pair programming in terms of code quality, we see that out of the data collected 66.6% of the developers feel that pair programming is effective in improving code quality. From the data collected we see that pair programming leads to nearly 15% more test cases passed which is consistent from the result in the existing publication – The Cost and Benefits of Pair Programming. We have also seen that pair programming leads to lesser LOC leading to better maintainability and readability. Thus for the first goal we can conclude that pair programming is effective in improving code quality to a great extent.

2. For the second goal to evaluate effectiveness of pair programming in terms of productivity we can conclude from the data obtained from the responses of the developers through online forums and results of existing publications that pair programming is helpful in delivering the project faster by completing it in about 25% to 40% lesser time. It also leads to lesser bugs and less time and money is spent on correcting bugs, thus making it more time and cost effective.

3. For the third goal to evaluate effectiveness of pair programming in terms of employee satisfaction we see from the data obtained that 60% of the developers like working in pairs, thus indicates majority of programmers feel satisfied working in pairs. Thus conclusion is also validated from the results in the publication – The Cost and Benefits of Pair Programming.

4. For the fourth goal to evaluate expertise level of both the programmers and nature of development task for successful implementation of pair programming, we see that in majority developers feel that the expertise level should be similar for better implementation of pair programming. And for the tasks, pair programming should not be used for every task and it works best for complex problems and finding errors. The conclusions have significant similarity from the results in the publications – Pair programming in software development teams – An empirical study of its benefits

Apart from all these conclusions based on collected data and referred publications, from my understanding there’s nothing good or bad with Pair Programming itself,
it’s just a matter of finding the right combination of developers for the right task. For that experience is required.

**Limitations**

Sometimes gathering data about certain aspects was difficult as developers themselves were not sure about the impacts of pair programming. So here I have a complete analysis of pair programming in my research and formulating questions for developers was challenging. The questions have to be concise and understandable. Sometimes it was difficult to include every scenario so the most important options were included. Like in the question - Q: What type of tasks should pair programming be used for? The four most options were included. The authenticity of the data relies on the response of developers as data is obtained and analyzed through their responses. The data was limited to responses from developers as it was difficult to visit organizations and conduct experiments there. I had limited time of nearly two months for data collection and I tried to collect maximum data to answer questions in the four goals for the project.

**Reflection:**

I enjoyed working on this project and gathered a lot of data about different aspects related to Pair Programming.

Next time I would like to include certain things in my project as I would like to study about what impact does the pair programming techniques has on novice programmers after the project is over.

I was not able to include this parameter as it was difficult to differentiate novice programmers from senior programmers. I would like to study that what impact does the techniques of pair programming have on novice programmers after the task is over. Do they learn when paired up with a senior programmer or they simply get the work done by doing minor tasks and do not learn much from the task.

I would like to visit organizations and conduct experiments there to study pair programming more effectively. I would also like to study more about pair programming in comparison to side programming as whether we can substitute pair programming with side programming.
References

Basic Information and Definitions:

https://www.google.com/#q=pair+programming+wikipedia
https://raygun.com/blog/2015/11/how-good-is-pair-programming-really/

Publications referred:

- Pair programming in software development teams – An empirical study of its benefits. (February 2008)
- The effectiveness of pair programming: A meta-analysis (July 2009)
- Strengthening the Case for Pair Programming (July/Aug 2000)

Data Collection – Google forms, Quora, Stack Overflow

List of Figures

1. Programmer preference on pair programming based on code quality .................................................. 7
2. Fellow friends’ preference on pair programming based on code quality ........................................... 8
3. Programmers view on quality ........................................................................................................... 9
4. Test cases passed analysis ............................................................................................................. 10
5. Post development test cases passed ............................................................................................... 11
6. Lines of code .................................................................................................................................. 11
7. Programmers review on productivity ............................................................................................ 14
8. Developers on effort reduction with pair programming ..................................................................... 13
9. Developers on project completion with pair programming ............................................................ 14
10. Relative time ................................................................................................................................. 15
11. Project completion time ................................................................................................................ 15
12. Developer preference on pair programming .................................................................................. 16
13. Enjor work more because of pair programming ........................................................................... 17
14. Expertise level ............................................................................................................................. 17
15. Task preference rating .................................................................................................................... 18
16. Partner useful for which task ......................................................................................................... 19