



# International Journal of Emerging Technologies in Computational and Applied Sciences (IJETCAS)

[www.iasir.net](http://www.iasir.net)

## Optimizing Pair Programming Practice through PPPA

Smitha Madhukar

Department of Computer Science / Assistant Professor  
Amrita Vishwa Vidyapeetham, Mysore Campus,  
Karnataka, INDIA.

**Abstract:** Pair programming is a technique by which two individuals collaborate to accomplish a single programming task. It is a good practice towards agile software development process. Pair programming increases the quality of the software product and achieves higher competitive benefits as compared to solo programming. Pair programming occurs when two programmers engage themselves in the development of software simultaneously at the same workstation. This results in lesser risks, higher productivity, enhances inter communication skills and improvement of technical expertise. This paper explores the benefits of pair programming and highlights the findings of a programming activity performed by students in a java lab. Furthermore, this paper has made a sincere attempt to present an analysis and has compiled the outcomes of pair programming methodology adopted in the lab with students as participants.

**Keywords:** Pair programming, cohesion, coupling, Driver, Navigator, Cyclomatic complexity, PPPA.

### I. Introduction

Pair programming is a software development strategy through which two programmers strive to complete a task by working at the same machine. The two programmers essay the roles of driver and navigator respectively. The driver creates the code and navigator tests the code as and when it is written. This type of programming practice scores over other methods because the navigator suggests many ideas for improving the quality of the code and helps in resolving many issues that may cause potential threats to the software. With the help of pair programming, the driver gets the freedom to concentrate only on the coding aspects and navigator primarily focuses on safety, reliability and performance of the end product. They keep swapping their roles quite often.

Pair programming is quite beneficial in enhancing communication between driver and navigator. Both help each other in eliminating coding deficiencies by making their ideas get implemented and constantly involve in exchanging technical views. This often results in the production of software high on performance. There are sufficient evidences to prove that when programmers work in pairs, they tend to produce substantially good results when compared to working solo. This form of programming claims to have more advantages in solving big problems.

With pair programming, two individuals effectively operate on the same algorithm, similar code and share equal credit when the logic succeeds. The concept of pair programming is inevitable in a software industry where professionals rely upon productive teamwork coupled with proficient technical synergy.

An almost analogous scenario prevails in an academic environment in which the same protocol can be applied to evaluate students on certain parameters like: performance, conduct, perseverance and persistence.

It can be noted that students performed better while executing java programs when they worked in pairs. They comprehend, reason and analyze well in a pair programming framework.

### II. Methods in pair programming

Pair programming can be adopted in the following forms:

1. Master-Master combination: This happens to be the best and proven solution for achieving good results.
2. Master-Amateur combination: This pairing leads to monitoring of amateurs by experts. It provides lot of Opportunities for amateurs to learn and practise novel approaches.
3. Amateur-Amateur combination: This combination is considered to be the least beneficial in terms of quality. However, this combination is strongly criticized in an industrial setup but followed in an academic context.

Remote pair programming is another variation to pair programming when driver and navigator are in different places. They work by means of desktop sharing or collaborative editors.

### III. Feasibility of pair programming

From an industry perspective, pair programming is found to be extremely helpful in saving the time and effort of a software professional. This programming practice creates lesser bugs, more readable code and ultimately an efficient architecture. Working collaboratively is a step towards developing reusable code.

Additional benefits are:

1. Pair programming helps in establishing better communication between professionals who take up the roles of driver and navigator.
2. Greatly reduces time and effort spent in coding and testing.
3. Since pair programming offers scope for switching roles between driver and navigator, it helps both to learn and implement new things.
4. In a software industry, when two experts with versatile experience are involved in solving a programming task, it becomes more simpler in finding a solution.
5. Pair programming introduces a higher degree of dynamic challenges.

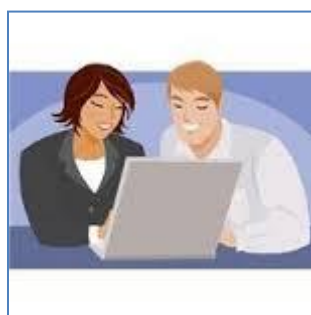


**Figure 1: Two professionals involved in pair programming**

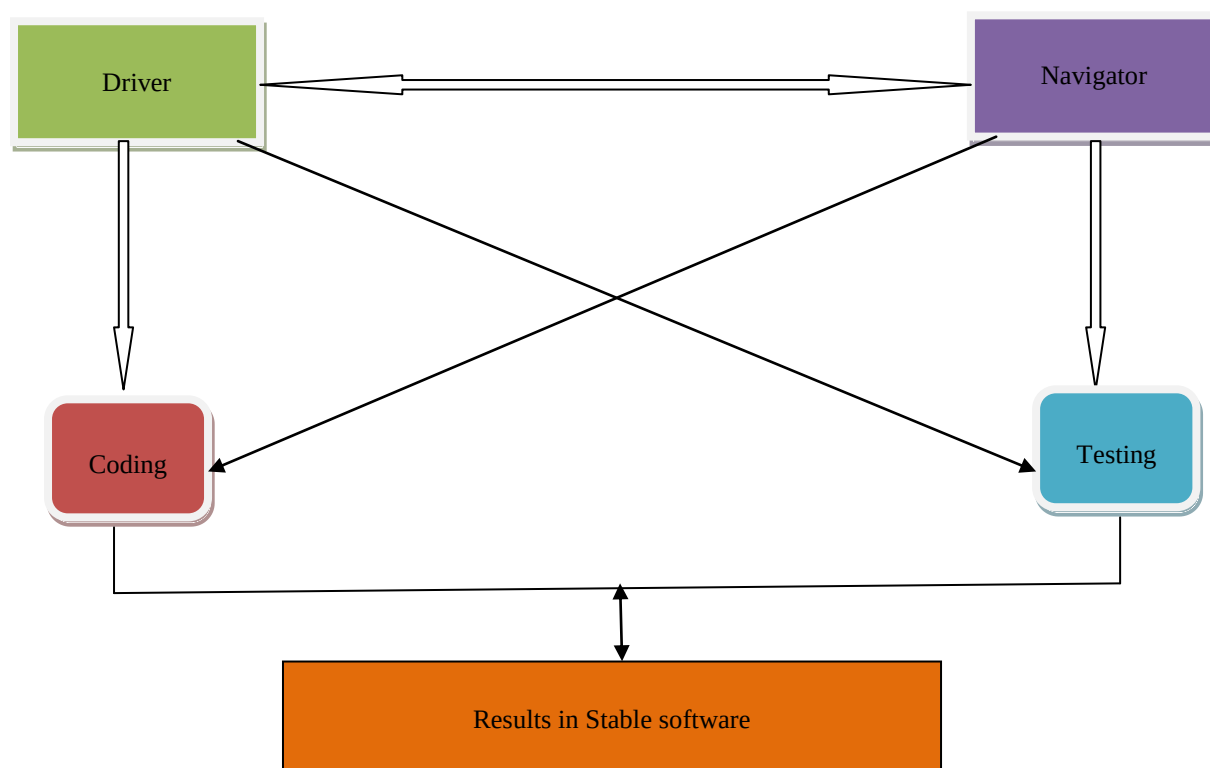


**Figure 2: Pair programming process**

#### **IV. Experimenting with Pair programming in Academic framework**

Computer Science happens to be a challenging discipline. Even more challenging is the manner in which it can be taught to students so that they can enjoy the experience and learn more. Therefore, computer science tutors are

always in the process of exploring innovative techniques to make teaching more effective and enriching. In this regard, an experiment was conducted wrt pair programming in java lab.

The details are as follows:

12 lab sessions were handled by the author to teach Java programming for postgraduate students. Students were asked to work on completing 25 programs related to advanced java concepts. The method of pair programming was applied and results were recorded by the author. It was compared to solo programming method previously adopted by the author to teach C++ to the same batch of postgraduate students in their previous semester. The purpose of this study was to examine the behaviour and performance of students during the lab sessions. Assessment was made based upon five key parameters.

A brief description of the outcome is presented below:

| Criteria           | Solo Programming( C++) | Pair programming( Java) |
|--------------------|------------------------|-------------------------|
| Participation      | 30%                    | 80 - 90%                |
| Inquisitive nature | 15%                    | 50 - 60%                |
| Behaviour          | 50%                    | 75%                     |
| Debugging skills   | 50%                    | 90 - 95%                |
| Perseverance       | 10%                    | 60 - 70%                |

**Table 1: Comparison between Solo and Pair programming**

In addition to enjoying their lab sessions, students also exhibited confidence and showed more interest towards learning more. There was a significant level of interaction between students that resulted in better quality of programs. From the above table, it can be inferred that, students performed rather productively when paired than solo programming.

Students were asked to answer a set of questions on the last day of lab sessions. The resulting graph is shown below.

### V. Analysis of PP Performance algorithm (PPPA)

In this paper, an algorithm called as Pair programming performance algorithm is presented to assess the results of pair programming efforts. By analysing this algorithm, it is easy to understand how pair programming has performed in the above mentioned academic setup. The factors taken into consideration for judging the performance are mentioned below:

1. Effort estimation
2. Time estimation
3. Cohesion
4. Coupling
5. Cyclomatic complexity
6. Bugs per line of code

The above metrics are measurable and can be quantified.

Let us examine the definitions of the above terms wrt. Pair programming methodology.

Effort can be defined as the combined endeavour of both driver and navigator in accomplishing the given programming activity.

Time can be defined as the total time taken by both driver and navigator in accomplishing the given programming activity.

Cohesion can be defined as the extent to which the unit coded by driver can be seamlessly integrated by the navigator when they switch between roles.

Coupling can be defined as the extent of dependency between components developed by driver and navigator.

Cyclomatic complexity is defined as the complexity of the developed code.

Bugs refer to something that the software does but it is not supposed to do.

Let effort be denoted by 'e'.  $e = \frac{1}{2} (\text{driver}) + \frac{1}{2} (\text{navigator})$ . Let 't' denote the time.  $t = \frac{1}{2} (\text{driver}) + \frac{1}{2} (\text{navigator})$ .

Pair programming proves the fact that when developers work in pairs, they seem to be quite fast in developing the code than single developer. It is quite evident from the above equation.

The coupling and cohesion estimates can be made as follows:

The percentage of seamless integration performed is found to be higher in PPPA because driver and navigator switch roles quite often. It introduces higher cohesion because the driver is constantly forced to convince the navigator before proceeding further. Quite obviously, when navigator takes upon the role of driver, then the navigator's developed module integrates seamlessly into the already existing modules of driver.

Let  $S_1$  be the set comprising of modules developed by the driver.  $S_1 = (M1, M2, M3, \dots, Mn)$   
 Let  $S_2$  be the set comprising of modules developed by the navigator.  $S_2 = (M1, M2, M3, \dots, Mn)$

The cohesion factor  $CF = (S_1 \cup S_2)$ .

The software, on the whole can work properly only when there is an average level of interdependence between the Programs of driver and navigator because it yields reusability. This is illustrated by coupling activity. To achieve higher rate of success in programming, it is noted that driver writes the code only when the navigator consents for it. The navigator certifies that the code is error free only when he applies various levels of tests and checks.

In PPPA, the rate of coupling is weak because both driver and navigator are striving hard to achieve the desired level of efficiency. Coupling measure (CM) can be interpreted as follows:

Since there is a possibility of one module invoking another, the inputs supplied to a module may be the resultant output of another module.

Let 'p' and 'q' be two modules interdependent. 'm' be the number of interconnections between them.

Let 'n' be the number of inputs supplied to the modules.

$$CM_{(p,q)} = \frac{p(n) + q(n)}{\Sigma n + \Sigma m}$$

Since the CM is relatively low and CF is high, it results in a comparatively reliable software. Cyclomatic complexity, CC is a quantitative measure to a function's complexity. The independent paths in a software developed using pair programming are relatively less. These proven results indicate that higher CC will lead to many errors. Therefore the number of bugs is dependent upon Cyclomatic complexity.

Let 'b' denote the density of bugs. Higher the CC, higher the bug density.

$CC \propto b$ .

## VI. Empirical evidence for pair programming conducted in academic environment

From the above theoretical representations and mathematical equations, empirical evidence can be obtained for the experiment conducted in academic environment. The students engaged themselves in rigorous programming in java and made a sincere attempt to extend the functionalities of the existing programs by introducing greater amount of flexibility to the code. Five parameters were presented to the students in terms of questionnaires to collect answers for the following questions. The findings are illustrated below:

1. What is the degree of satisfaction obtained after your pair programming experience?
2. Did you experience resilient flow during pair programming?
3. Do you support the concept of collective code ownership during pairing?
4. Did u proceed with doing the right thing?
5. Were there fewer interruptions during pairing?

The below graph depicts the findings of the author based upon the answers given by the students in view of pair programming. The author interviewed each student to get deeper insights about the adopted strategy.

The degree of satisfaction was observed as high.

The flow during pair programming happens quite smoothly. Hardly susceptible to interruptions.

Students were of the opinion that they gained a good amount of expertise when they were subjected to pair programming. Eventhough they worked in pairs, they gained a better understanding of all the programs by exchanging their ideas with their partners.

Students requested for less number of breaks during lab sessions since they enjoyed the whole process.

Students opined that they encountered fewer interruptions compared to solo programming.

Thus the graph illustrated below justifies the pair programming performance algorithm.

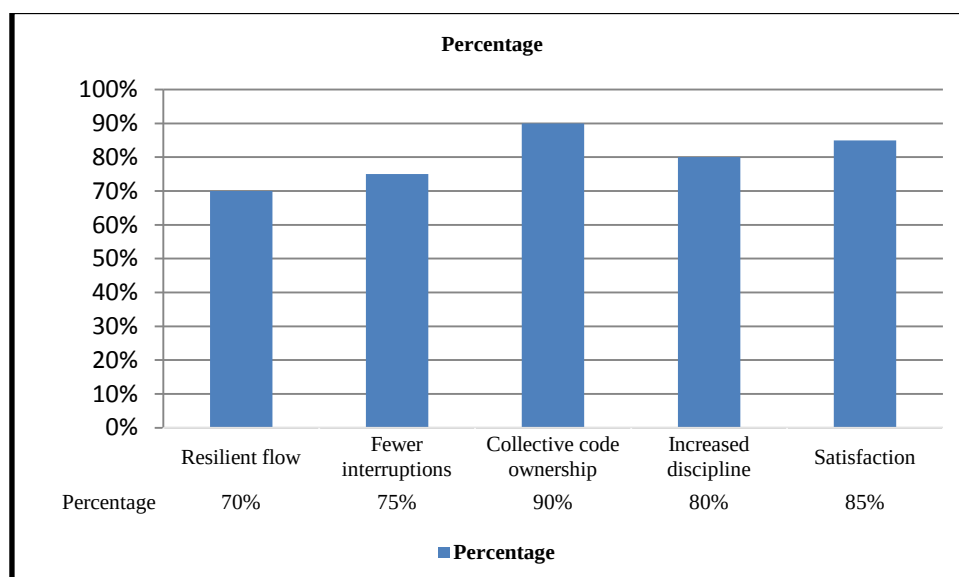


**Figure 3: Evaluation of Pair programming performance algorithm**

## VII. Conclusion

This paper has provided a broader view of pair programming by illustrating an example from the academic perspective. There is tremendous scope for future work to consider a live scenario from the software industry perspective. The experimental results provided forms as basis for enhancing this work. The same experiment can be extended to include more parameters. The primary advantage of PPPA is that it supports the methodology by presenting ample evidences. In the current context, the PPPA works fine with the existing factors mentioned above. It can be observed that PPPA shows a significant improvement in analysing pair programming techniques.

## REFERENCES

- [1] Laurie Williams, Strengthening the case of pair programming, 2000.
- [2] Salleh, A systematic review of pair programming, 2008.
- [3] Williams, Guidelines for implementing pair programming, 2006.
- [4] Gallis. H, An initial framework for pair programming, 2003.
- [5] Roland Doepke, Markus Soworke, Pair programming interfaces and research, 2009
- [6] Tammy Vandergrift, Coupling pair programming and writing, 2004
- [8] Sven Heiberg, Unno Puus, Priit Salumma, Asko Seeba, Pair programming effect on programmer's Productivity, 2003
- [9] Mawarny Rejab, Mazni Omar, Mazda Mohd, Kairul Bahiya Ahmed, Pair programming in inducing Knowledge sharing, Proceedings of the 3<sup>rd</sup> international conference on computing and informatics, 2011.
- [10] Mark Antony Poff, Pair programming to facilitate training of newly hired professionals, Mark Antony Poff, thesis submitted to Florida Institute of Technology, 2003.