Solo Programming and Pair Programming in Software Metrics

Arpita Mittal
Sanjay Kumar Dubey

Abstract

In the software development environment Software metrics is an attribute (consisting of a quality, or characteristic of an object) derived from the measuring attributes of software objects. Software Metrics is termed as the measurement based techniques for software development processes and products to supply meaningful and timely management information, and the use of those techniques to improve that process and products [GOOD93]. Metrics basically are used to improve software quality and productivity. Software metrics deal with software items. While dealing with software metrics various management activities are there which are discussed in the latter part of this paper. Solo Programming is where one programmer develops software alone. Pair Programming is where programmers develop software side by side at one computer. Pair Programming is basically referred as Collaborative Programming[2]. While dealing with software metrics sometime some problem of pair programming (also termed as team programming) and solo programming occurs. In this paper we compared solo programming and pair programming, and listed the benefits and defects of both programming (pair programming and solo programming) that occur while working with teams or without in software metrics.

Keywords Used: Pair Programming, Solo Programming, Collaborative Programming, Code Reviews and People Factors.

1 Introduction: Software Engineering is used in the development of all kinds of software systems from software system initial phase of development, testing, maintenance and usage phase[1]. Software Metrics are very essential aspects of software development from its early phases of development to achieve maximum benefit and increased productivity. Different companies have different requirements for software development. Also even in the same company there will be different requirements, cost, scope for different projects[1]. So in order to do this we think in the context of Solo Programming And Pair Programming.

Software Metrics: The term software metrics have no scientific meaning, it is a made up term. Software metrics is used to define the measure of software items.

Assistant Professor, Department of Computer Science, MIET Meerut, U.P., INDIA.
Assistant Professor, Department of Computer Science, MIET Meerut, U.P., INDIA.

TABULARFORM

Software items	Software metrics activities	Software Metrics Qualities	Software Metrics Requirements	Software Metrics Uses
1.A part of production of a software product	1. Cost and effort estimation	1.Should be simple and precise	1.Improving visibility.	1.Macro estimating
2.An event such as production failure	2.Data Collection	2. Should clearly specify its objectives	2. For improving planning and control	2.Micro estimating
3.A person involved in software production.	3.Quality models and their measure.	3.Should be easily obtainable and affordable	3. For improving quality	3.Allocation of resources
4.An organization.	4.Productivity models and measures	4.Should be valid.	4. For improving productivity.	4.Progress assessment
	5.Performance measures	5. Should be robust		5.Process improvement assessment
	6.Reliability models			6.Prediction of code completion
	7. Capability-maturity assessment			7.Defect prediction
	8. Structural and complexity measures			8.Process improvement
	9.Management measures			9.Project improvement
	10.Evaluation of methods and tools			10.Project re-planning

Pair Programming: Pair Programming is a programming where two people or programmers sit side by side and work collaboratively on the same project. This practice has improved the way of developing software over the last two decades. In pair programming basically two people work to develop a code.

Solo Programming: Solo Programming is a programming where one person sits on a system to develop a project. This practice only one person sits to develop a code.

2.Problem Statement : When working with software metrics sometime team conflicts may occur due to different views of the different programmers. Since Software metrics as discussed above is necessary just because for improving visibility, for planning and control, for improving quality and for improving productivity. So sometimes when working with software metrics there occurs many issues when working in a team (pair programming). Also when working in solo programming in software metrics we do not be able to have proper understanding and can't be able to get different views related to the software project.

Sometimes person prefer to work in pairs and the term is termed as pair programming and sometimes there is necessity for working in solo environment. So, in this paper we have selected the metrics and the metrics of both the types of programming (pair programming and solo programming)

Both types of programming whether it is solo programming or pair programming both have its advantages and their disadvantages when working with software metrics, especially when calculating cost and effort (when measuring manager's productivity) estimation in software metrics.

3. Solo programming and pair programming

Pair Programming: Pair programming have been taught from the early stages of development and is practiced as a solitary activity [2][3]. Managers view pair programming as a scarce resource and many other experienced programmers does not want to work with other programmers due to their own reasons while some programmers want to work in pairs because such programmers think that working in pairs will the work done faster, resulting in simple code, that is easier to understand.

This raises some provocative questions that Is pair programming is really more effective than solo programming? What are the economics? What about the people factor- enjoyment of the job?
Based on the interest in pair programming and solo programming this paper examine the results and has given beneficial factor of both types of programming.

Solo Programming: Solo Programming is a programming [3] where developer and coder works separately alone to develop software. Single programmer knows what is the program specification, as to what software develop, what to design, code, reviews, testing and maintenance etc.

Solo programming is advantageous where programmers wanted to work alone and doesn't want to work in the team to develop the software due to some of their reasons.
Some managers think pair programming as reluctant .

This raises the question Is solo programming more effective? This paper raises the questions and has told about the beneficial factor of both types of programming.

4. Comparision

TABULAR FORM

Pair Programming	Solo Programming
Method: New	**Method: Have been used from many years. It is traditional development of software method**
Advantages	**Advantages**
1. Knowledge transfer.	**1.** In Solo programming as programmer and developer work alone, so neither programmers nor developer have to wait for the others, so solo programming results in effective time management.
2. Higher design quality having fewer defects.	**2.** Those who are not comfortable while working in pairs can work alone.
3. Higher satisfaction and confidence.	
4. Less time to complete the task.	

Pair Programming	Solo Programming
Disadvantages	**Disadvantages**
1. Some people received undeserved credit while working in pair programming.	**1.** Since in solo programming work is done alone, so ideas cannot be shared, no sharing of ideas occurs in solo programming.
2. Due to pair programming disagreement occurs among the persons in pair programming	**2.** Since developers and programmers work alone so solo programming is time consuming.
3. Pair Programming increases the cost as the number of persons gets increased in pair programming.	**3.** While working in solo programming misconstruing of tasks can occur.
4. Time got scheduled in pair programming.	

5. Conclusions Both Solo programming and pair programming have their own benefits and disadvantages. Solo programming is beneficial when the developer and designer wanted to work alone, no one wanted to be there to share the undeserved credit, and doesn't want to be in some types of disagreement that occur while work is to be done in a team, also doesn't wanted to increase the cost of the project by unnecessarily involving more persons in the project. While team management occurs in pair programming. In pair programming time gets reduced as a team work for the development of the project, resulting in sharing of ideas, knowledge transfer etc.

6 Future scope: In the nearby future we can see the more efficient technologies and term selected software meters which working in pair programming or solo programming. We can also use the techniques for like neural network, fuzzy logic and genetic algorithm. Also we can work with agile methodology, which most recent emerging technology these days. When we are working for improving of software quality all of these technologies are providing many good solutions.
Neural networks are basically concerned with dealing the neural structure. Neural networks basically concerned with the neurons.

Fuzzy logic is the science of dealing with prediction. Fuzzy logic deals with approximate a value which in contrast with crisp logic which only deals with true and false values.

Genetic Algorithm is basically a heuristic search technique which provides the optimal solution of the problem domain it doesn't guarantee to provide the best solution of the problem domain.

The future work can include Genetic Algorithm Technique Comparison with other techniques used for fault and maintainability.

Hence, when software meters work with any of these technologies, work well and provide better results in improving the quality of software.

References

1. Understanding Tools and Practices for Distributed Pair Programming-Till Schümmer (FernUniversität in Hagen, Department for Mathematics and Computer Science, Cooperative Systems, Germany till.schuemmer@fernuni-hagen.de) Stephan Lukosch (Delft University of Technology, Faculty of Technology, Policy, and Management, Systems Engineering Section, The Netherlands s.g.lukosch@tudelft.nl)

2. The Costs and Benefits Of Pair Programming-Alistair Cockburn Humans and Technology 7691 Dell Rd Salt Lake City, UT 84121, USA arc@acm.org 801.947.9277, Laurie Williams University of Utah Computer Science 50 S. Central Campus #3190 Salt Lake City, UT 84112, USA lwilliam@cs.utah.edu 435.649.7931.

3. New Metric for Measuring Programmer Productivity-Marwa Solla1, Ahmed Patel2, Christopher Wills3 1,2 Department of Computer Science, Faculty of Information Science and Technology Universiti Kebangsaan Malaysia 43600 Bangi, Selangor Darul Ehsan, Malaysia 1 Department of Computer Science, Faculty of Sciences Al Fateh University, Tripoli, Libya 2,3 Faculty of Computing Information Systems and Mathematics Kingston University, Penrhyn Road, Kingston upon Thames KT1 2EE, United Kingdom 1marwa solla *at* gmail.com,

2whinchal2010 *at* gmail.com, 3ccwsl *at* kingston.ac.uk.

4. Sommerville, "SOFTWARE ENGINEERING 9," *Inc, publishing as Addison-Wesley,* 2009.
5. A.Patel *et al.*, "Ahmed Patel, Liu Na, Rodziah Latih, Christopher Wills, Zarina Shukur and Rabia Mulla," Journal of Computer Science, vol 6 No 11, pp.1406-1415, 2010 doi 10.3844/jcssp.2010.1406.1415[3] L. Williams and R.Kessler, "Pair Programming Illuminated," *Addison Wesley, July* 2002, 288 pages, ISBN 0-201-74576-3.
6. J.E.Hannay *et al.*, "The effectiveness of pair programming: A meta-analysis," *ScienceDirect, July* 2009. vol 51, pp. 1110-1122 DOI 10.1016/j.infsof.2009.02.001.
7. J.T.Nosek, "The Case for Collaborative Programming," *ACM, March* 1998, vol 41 issue No 3, pp.105-108, doi 10.1145/272287.272333.
8. L.Williams *et al.*, "Strengthening the Case for Pair Programming," *IEEE Software, July* 2000, Vol 17 NO 4, doi 10.1109/52.854064.
9. A.Cockburn and L.Williams, "The Costs and Benefits of Pair Programming," *presented at extreme Programming and Flexible Processes in Software Engineering XP 2000, Cagliari, Sardinia, Italy, 2000* doi 10.1.1.26.9064.
10. H.Hulkko and P.Abrahamsson, "A Multiple Case Study on the Impact of Pair Programming on Product Quality," *IEEE International Conference on Software engineering,* pp.495-504, 2005, doi 10.1145/1062455.1062545.
11. J.Vanhanen and C.Lassenius, "Effects of Pair Programming at the Development Team Level: An Experiment," *IEEE,* 2005. pp. 336-345. doi 10.1109/ISESE.2005.1541842.
12. J.Nawrocki and A.Wojciechowski, "Experimental Evaluation of Pair Programming," *proc. European Software Control and Metrics (Escom)* 2001. doi 10.1.1.9.1689.
13. E.Arisholm *et al.*, "Evaluating Pair Programming with Respect to System Complexity and Programmer Expertise," *IEEE Computer Society,* 2007. VOL 33, NO 2, doi 10.1109/TSE.2007.17.
14. R.Gunnalan *et al.*, "Pseudo Dynamic Metrics," *IEEE Computer Society,* 2005, ISBN 0-7803-8735-X.
15. N.E.Fenton and M.Neil, "Software metrics: successes, failures and new directions," *The Journal of Systems and Software 47 (1999) 149-157,* 1999.
16. G.Sikka *, et al.*, "Estimating Function points: Using Machine Learning and Regression Models," *2nd International Conference on Education Technology and Computer (ICETC)* 2010, doi 10.1109/ICETC.2010.5529600.
17. D.J.Ram and S.V.G.K.Raju, "Object Oriented Design Function Points," *Quality Software, 2000. Proceedings. First Asia-Pacific Conference IEEE,* pp.121-126, 2000, DOI 10.1109/APAQ.2000.883785.
18 T.Hongwei *et al.*, "The Research on Software Metrics and Software Complexity Metrics," *IEEE International Conference on Cognitive Informatics,* 2009, pp. 131-136, doi 10.1109/IFCSTA.2009.39.

19. S.R. Chidamber and C.F. Kemerer, "A Metrics Suite for Object Oriented Design," *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 22, Issue No. 6,* 1994. doi 10.1109/32.295895.

20 R. Dillibabu and K. Krishnaiah, "Cost estimation of a software product using COCOMO II.2000 model – a case study," *Science Direct,* 2005. Vol. 23, Issue 4, May 2005, pp. 297-307, doi 10.1016/j.ijproman.2004.11.003.

21. K.M. Lui and K.C.C. Chan, "Pair programming productivity: Novice–novice vs. expert–expert," *Science Direct,* 2006, Vol. 64 Issue 9, September 2006. doi 10.1016/j.ijhcs.2006.04.010.

22. C. McDowell *et al.*, "The impact of pair programming on student performance, perception and persistence," *International Conference on Software Engineering, IEEE Computer Society,* 2003. http://doi.ieeecomputersociety.org/10.1109/ICSE.2003.1201243.

23 T. Bipp *et al.*, "Pair programming in software development teams – An empirical study of its benefits," *Science Direct,* 2008. Vol. 50, Issue 3. doi 10.1016/j.infsof.2007.05.006.

24. L.F. Johnson, "On Measuring Programmer Team Productivity," *Electrical and Computer Engineering, IEEE Canadian Conference,* vol. vol. 2, pp. 701-705, 1998. DOI 10.1109/CCECE.1998.685594182.

25. Pressman-Software Engineering [Second Edition]