GSOFT: A Generic Model for Teaching and Learning Large-scale Software Programs

Radhakrishnan .P
Department of Computer Science & Engineering
Pondicherry Engineering College
Puducherry

Kanmani .S
Department of Information Technology
Pondicherry Engineering College
Puducherry

ABSTRACT

Learning software engineering in practical laboratory based on present curriculum is under the clutches of hard practices and not well defined. The effort and time taken for planning, designing and coding, have a few issues in student's learning aspect. Same is the case for evaluating the student's program. With the effect, both the teacher and students have leaned towards lightweight learning method using Extreme Programming (XP). Pair Programming, which is one among the twelve practices of XP has been widely used by the pedagogical community. However, this practice is found suitable for introductory level small-scale programs. Also, the researchers incorporating pair programming, have not given much importance for program assignments and program correction. To address these problems, we have developed a Generic software teaching and learning model called GSOFT with few software development practices, and pair programming. COSMIC FFP (Common Software Measurement Integration Consortium Full Function Point) standard was used for program assignments and program evaluation. This method was applied on real time project assigned to student as large scale programs and examined. For this study, the students were grouped into pair programmers (PP) and solo programmers (SP). The performance of the PP and SP groups were measured using our generic model and found the person-days taken to complete the program. The results show that PP used less person-day than the SP. This study also proves that the program developed by PP has better coding.

Keywords

Pair Programming, Extreme Programming, Software Measurement, COSMIC FFP, Cfsu, GSOFT, Person-days.

1. INTRODUCTION

Traditional teaching and learning methodologies are too weighty for the rapidly changing educational requirements. Extreme Programming [2] is a light weight software development model, gained support for its widely accepted practices, for developing Software. The practices are

- Planning game
- Small releases
- Metaphor
- Simple design
- Testing
- Refactoring
- Continuous Integration
- Collective ownership
- Pair programming
- 40-hour week
- On-site customer
- Coding standards

Several researchers have integrated these practices into the curriculum of computer science and software engineering programs[8]. They present excess benefits for the students while learning to develop software in practical study. The general practice of program development by the students and program correcting by the teachers is shown in fig. 1. Not all the practices drive the students towards cognitive domain, since some are hard to understand (e.g., refactoring) and some are hard to follow (e.g., 40- hours a week and collective ownership). However, the practice of pair programming is generic and plays a pivotal role in which all other practices spin around, for the successful application. When XP practices are applied through pair programming in the laboratory based learning, the students create stronger, more powerful experience and accelerates the learning process.[14] as they are more prone to collaborative and interactive process. In practical learning, it is a formal expectation by the students that their assignments are to be evaluated for its overall output by means of some effective methodology. Several experiments/study involving students with PP used different method to evaluate their performance and their problem solving ability. However, those studies generally focus to learn about the merits and demerits of pair programming, and none of the study gave importance for correcting the student's work, which is a valid predictor to measure the effects of pair programming. In the software industry, the programs are being measured and evaluated through various methods. The latest and widely used one is COSMIC-FFP (Common Software Measurement International Consortium, Full Function Point), which provides a standardized method of measuring the functional size of software in accordance with Functional User Requirement (FUR) in the business perspective. Now, the issue is, "Whether this perspective could be applied to pedagogical domain while teaching and learning and whether the student's program could be measured using COSMIC FFP standards?" To address this question, this study extracted the essence of few practices from XP and COSMIC FFP standards, developed a generic software teaching and learning model called GSOFT as an alternate method, which was extensively followed by the pair programming students.

Further, the study is organized as follows. Section 2 surveys the literature. Section 3 define and list the Study design. In section 4 the case studies used and the result are discussed. Finally, section 5 conclude the study and discuss the future work.

2. LITERATURE STUDY

The literature study considered two feature. Pair programming for teaching and learning perspective and COSMIC FFP method for assigning task with equal in nature

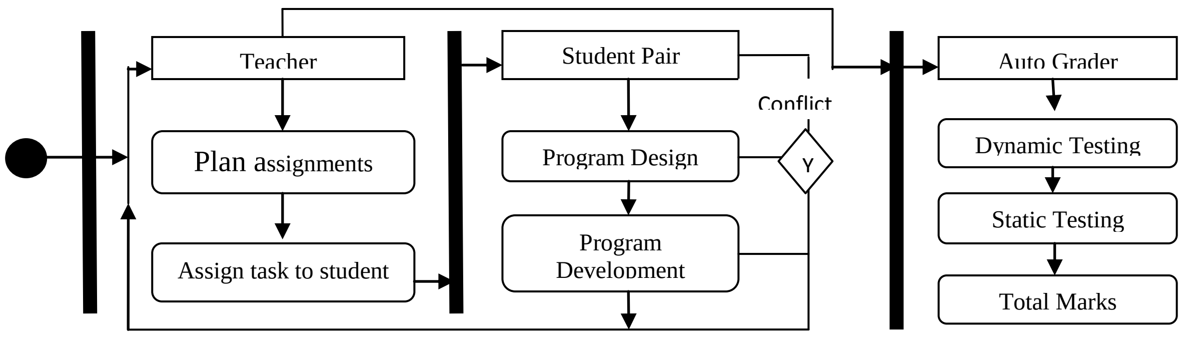

Fig. 1. Collaborative teaching and learning activities by students and teacher

to the students, and evaluating them in the perspective of correctness and the person-hours to complete the assignments.

2.1. Pair Programming

Pair Programming involves two programmers working collaboratively on one computer, one as a driver who operates the keyboard, concentrates on the lower level details of the task and another as a navigator who observes the driver, offering suggestions and corrections on higher level details of the task [3]. In contrary, SP is the traditional method, developing software program individually. The Solo Programmer decides himself how the code is going to be, owns the code, with all the good and bad, associated with that program. Several previous controlled experiments in programming industries claimed that pair programming is useful and beneficial in numerous facets [16] as well as in educational set ups [8]. Program correctness [7]; Higher software quality [14]; Reduced time for program development [17],[18], [4]; Increased learning efficiency [18]. Increased confidence level [3], [12]; Course completion rate [9],[3] are some of the widely gained benefits found in the literature. Apart from the benefits of increased performance of the students as listed above, PP practice is also proved to be beneficial in other human centric factors such as motivation, general ability and expertise [8] and increases personality traits of the students [1],[8],[12],[15].

2.2. COSMIC FFP

The COSMIC FFP [6] Method defines a standardized measure of Software Functional Size expressed in CFP (COSMIC Function Points) units. The software is measured by the level of decomposition and the level of granularity. The COSMIC generic model suggests four different types of data movement in a software program, such as Entry; Exit; Read; and Write. Entry type move data across the boundary from the user to the functional process. Exit type move data across the boundary to the user. Read type move data from persistent storage in the functional process. Write type move data from the functional process to persistent storage. Each elementary data movement constructs one Cfsu (COSMIC function size unit) [5].

3. STUDY DESIGN

This study is designed to evaluate the effort put in by the students while learning to develop software program as practical study using our Generic Software Development Model. For that reason, the following research questions were framed.

Question 1: Does the student using our generic software development model creates better coding with less duration?

Question 2: Does the Pair Programmer outperform the Solo Programmer in the required person- day while learning and developing software program, using our generic software development model?

3.1. Generic Software Teaching Model (GSOFT)

GSOFT is a deliberate and disciplined approach to software teaching and learning GSDM consists of Practices, Planning, Measurement, Effort Estimation, and Construction, for which the essence is derived from Extreme Programming (XP) and COSMIC Full Function Points and then integrated. GSOFT aims towards teamwork and implements a simple and effective way to enable groupware style of program learning, where students communicate with their teachers and fellow students frequently and freely. This improves the student's skill in communication; simplicity; feedback; and courage. The students develop their software design in a simple and clean manner. They get feedback from the teacher by testing their program repeatedly, starting from the scratch. Code development and frequent testing help the students to toss and rewrite the code if the teacher does not approve it. Repetition improves the scope of coding standard. The Important practices followed in GSOFT are discussed below:

3.1.1 GSOFT Practices

The GSOFT consists four core practices, extracted from XP [19] and COSMIC – FFP [20] model.

1. *PAIRING :* Learning alone in a conditioned set up in the lab is sceptical to progress. But in PP, they overcome from their poor programming practices quickly. In an experiment conducted by William et al [18], PP were good in completing their first assignment, but they took 60% more programming hours. In second assignments, they adjusted dramatically to a minimum of 15%. Pairing of students could be on different combinations such as expert with expert; expert to novice; novice to novice. Since PP increases the human centric behaviour of students, Pairing could also be "extrovert with introverts or heterogeneous with homogeneous kinds of students. Sometimes, incompatibility in pairing could limit the pair programmer's effectiveness [13]. To overcome, pair gelling facilitates the students to withdraw or change the pair at any stage.

2. *EQUAL TASK ASSIGNMENTS:* A common concern for the teacher is to bring up their students equally. Giving assignment and assessment to the students should be equal without any discrimination [10]. The algorithm for assigning the task equally is as follows.

```
Step1: Make User Stories (assignment) by the Teacher
Step2: Take one story at a time and repeat Step:4 to
        Step:5 until all the Stories are over
Step3: Split Story into Tasks
Step4: Find Cfsu for each task
Step5:Assign tasks of a story to students on their choice
        with equal Cfsu.
Step6: Coding by students adopting coding standards
Step7: Invoke GSOFT for evaluating the program.
```

3. *ON SITE TEACHING*: Teacher is readily available in the lab to answer the questions, resolve disputes in program development, and set guidelines for the students with priorities. The continuous correction and suggestion increases the student confident level and allow the student to proceed further without stagnating.

4. *CODING STANDARDS:* Coding standards are the set of guidelines for specific programming language. The guidelines include declarations, naming conventions, statements, method size and so on. The students should learn about the programming principles and programming rules to improve their source code and make software maintenance easier.

5. *CONTINUOUS EVALUATION:* Pair of students adapts the changing requirements in program development by more regimented flow into smaller groups of work as stories. In the process, these stories are prioritized, then the stories split into smaller groups of work (task). The on-site teacher evaluates each story immediately, which helps the students to proceed further with more confidence. This continuous evaluation, establish consistency in grading the students' work.

3.1.2 GSOFT Planning
GSOFT Planning is divided into two steps.

1. *MAKING STORIES*: Students have the right to get the most possible value out of every programming moment by asking for small atomic bits of functionality and the need for it.

2. *SYSTEM METAPHOR:* It helps to keep the design simple, clear and helps the student programmers to guess the needs to be done and how. When working without a clear metaphor (which is done in many projects), the students are expected to go for more diagrams, more design, more discussions, and more documents.

3.1.3 GSOFT Estimation
The purpose and scope is to size the functionality which corresponds to the effort interested in. For an application, written partly in Java and in VB, measurement is made separately for software size and development effort. First, identify the software layers for selecting the appropriate application language such as VB or Java or C++. Then identify the boundary, users, trigger, functional processes, data groups and sub-processes. In each of the above, identify the number of elementary data movements of entry, exit, read and write. Inside the completion of each module, the PP students plan with clear objectives for solid measurement. The XSOFT estimation is used to calculate the Cfsu of each task. One Cfsu is 5 elementary data movement of either "enter, exit, read or write".

$$\text{No. of Cfsu of a task} = \text{int}(\alpha+4)/5 + \text{int}(\beta+4)/5 + \text{int}(\gamma+4)/5 + \text{int}(\delta+4)/5$$

where α,β,γ & δ are number of enter, exit, read and write respectively.

No. of Cfsu of a Story = Total number of Cfsu of all the tasks in the story.

No. of Cfsu of the Project = Total number of Cfsu of all the stories of the project.

3.1.4 GSOFT Construction
The GSDM construction is integrating the task done by the student programmers into a solid application. The students take the tasks of equal size of their choice. Frequent testing and integration is done after completing each task. To check the efficiency by means of quality and quantity the details are recorded. The quantity as person-days is checked by means of the time taken to develop the number of Cfsu. The teacher, through code walk, checks the quality of the program statically.

4. CASE STUDIES AND DISCUSSION
For this study, two projects were selected as case study-1 and case study-2. Case study-1 is to examine the research question and case study-2 is for re-addressing the question as secondary proof.

4.1 Pre Processing
This study aims to evaluate the student's performance by inducting PP policy for program development and COSMIC FFP for program evaluation. Hence, determining the methods for pair construction (gelling) and Cfsu calculation for allotting the task equally to the students, as pre process were omnipotent.

4.1.1 Pairing
9 pair of students and 18 students as solo have been selected for this study A carefully developed procedure was followed in selecting the students with equal level of expertise in programming. Out of 142 students from B. tech 3rd year, only 36 top ranking students were selected based on their previous performance. The students were explained about the study and obtained their willingness to work. All the students expressed their willingness to work without any prejudice for 8 hours in a day. The participating students were divided into two equal groups as PP (Pair Programmer) and SP (Solo Programmer). Using the sorted array of the rank of the students, 9 pairs were formed for PP by joining first ranking with last ranking students iteratively. The remaining 18 students were marked for SP as shown in fig 2.

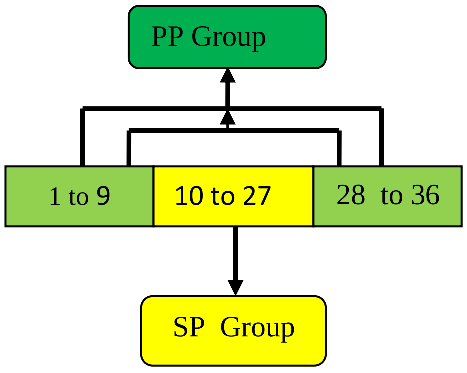

Figure 2: Scheme for selection of students for PP and SP.

4.1.2 Person day calculation

The person-day (one person-day=8 hours) calculation involves by splitting the project into stories, and stories into sub tasks. Using Xsoft measurement the Cfsu for each task was found. This process helps the teacher to assign 50% of the story to PP and 50% to SP equally with their choice.

4.2. Case study 1

For the first case study, a real time project namely "Mobile Park" was designed. The layer identified for this project is "Telecommunication" (TC). The project was split into 12 stories and prioritized. The resulting Cfsu for each story is listed in table 1. Since the total Cfsu in the first project were odd in number, 51 Cfsu were allotted to PP (ie. 6 Cfsu each for 8 pair and 3 Cfsu for the last pair) and the remaining 50 Cfsu were allotted to SP (ie. 3 Cfsu each for 14 SP and 2 Cfsu each for 4 SP) . The students have followed the Xsoft practices and developed the code. Teachers estimation was to complete the project in 6 days using 36 students. The PP group completed their tasks in 5 person- days, 7 SP group completed their task in 6 days and others completed in 8th day. The resultant data of the completion of task both by PP and SP in day wise are listed in the table 2.

Table 1 : Mobile Park- Task Development

Layers	Story Priorities	Cfsu
Telecommunication	Inward entry	14
	Outward exit	14
	Storage at	15
	Policy Entry	15
	Loading Credits	15
	Effected Parking	4
	Client Account	4
	Parking Policy Database	4
	Town Parking Tax	4
	Credit Purchase	4
	Upload	4
	Download	4
	Total	101

Table 2: Mobile Park-Task Completion and the Person- days between PP and SP.

DAY	PP	SP	CUMMULATIVE TOTAL	
			PP	SP
1	0*	0	0	0
2	0	0	0	0
3	2	0	2	0
4	6	1	8	1
5	10	4	18	5
6	0	5	18	10
7	0	3	18	13
8	0	5	18	18
Avg.	4.4 **	6.38	4.4	6.38

* Non completion

** Average = Σ (day * number of programmers completed their task) / Total number of programmers

Using the above data in table 2, a chart is drawn for clear view of the result (fig. 3). It indicates that, PP out performed SP. The data show that the PP took 4.4 days on average to complete 6 Cfsu whereas the SP took 6.38 days in average to complete 3 Cfsu. It is clear that the SP takes 0.45 times more to complete the task, when compared to PP(Table 3). The fig.3 shows the number of programmers completed the given tasks (3 Cfsu) on day 1 to day 8.

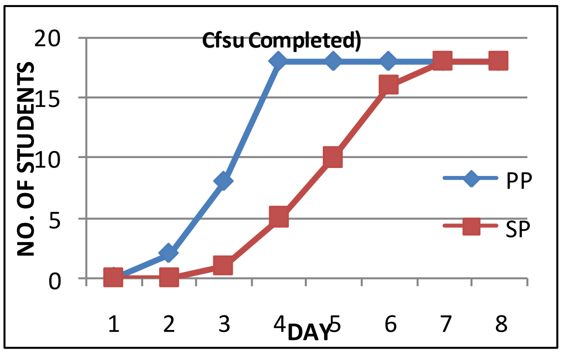

Fig. 3 : Mobile park- task completion by PP and SP

Table 3: Mobile Park -Person-Days -Comparison between PP and SP

Functional Process	Actual Size (Cfsu)	Allotted Size (Cfsu)		Actual Man Days (5 Cfsu)	
		PP	SP	PP	SP
City Entry	14	9	5	8	10
Zone Entry	15	10	5	8	10
Policy Entry	15	10	5	9	11
Loading Credits	15	10	5	8	11
Parking details	20	13	7	10	14
Upload & Download	8	5	3	5	7
AVERAGE	14.42	9.43	7	8.14	10.57

The fig. 4 shows the cumulative total number of programmers completed the given tasks (3cfsu) in day1 to day8

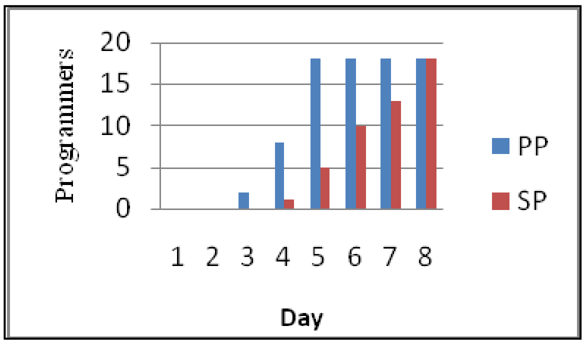

Fig. 4: Mobile Park -Cumulative number of task completion by PP and SP

4.3. Case study 2

The second, real time project namely "Shipping-Cargo" was designed by the authors. The layer identified in this project

Table 4: Shipping Cargo

DAY	PP	SP	CUMMULATIVE TOTAL	
			PP	SP
1		0	0	0
2	4	1	4	1
3	8	3	12	4
4	6	7	18	11
5	0	3	18	14
6	0	4	18	18
Avg.	3.11	4.33	3.11	4.33

was "Management Information System" (MIS) and it was developed using Visual Basic. The project was split into 12 stories and the resulting Cfsu for each story is listed in table 4. From the total Cfsu, 99 were allotted to PP (11 each for 9 PP) and the remaining 100 Cfsu were allotted to SP (6 each to 10 SP and 5 each for 8 SP). The teacher's expectation of person-days for completing this project by students are 4 person days with 36 students. But, all the PP completed their tasks within 4 person days, whereas only 11 SP completed their task in 4 person-days and others completed on sixth day. The data of the task completion is listed in the table 5.

Table 5 : Shipping Cargo -Task Completion with person day by PP and SP

Layer	Story Priorities	Cfsu
MIS	Shipment Details	29
	Consignor / Consignee Details	17
	Inbound/Outbound	22
	Report-Finance	22
	Report-Planning	10
	Booking Details	13
	Shipment Status	10
	Planning by Truck	10
	Status Report	21
	Shipment Sold	19
	Consignee Reverse	14
	Account Details	12
Total 199		

Using the data show in table 5. a chart is drawn for clear view of the result. It shows that, PP have once again out performed SP (Fig. 5) . The data in the table. 5 shows that the PP took 3.11 person days to complete 11 Cfsu whereas the SP took 6.38 person days to complete 6 Cfsu.

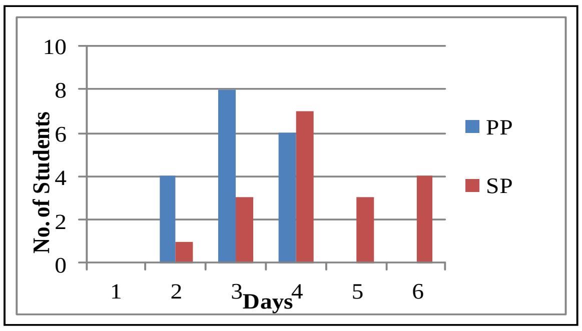

Fig. 5: Shipping Cargo -Task Completion PP and SP - MIS Project

The SP took 0.39 times more, when compare to PP for completing the tasks. The fig.6 shows the number of programmers completed the given tasks (3 Cfsu) on day 1 to day 6. The fig.6 shows the cumulative total number of programmer completed the given tasks (3 Cfsu) on day 1 to day 6.

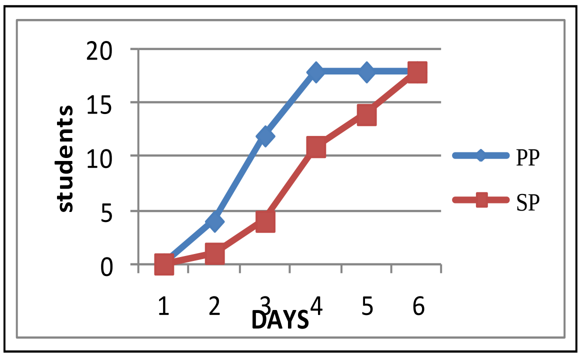

Fig.6: Cumulative total of performance by PP and SP

5. CONCLUSION AND FUTURE WORK

A new Generic Software Teaching model called GSOFT was developed by integrating few features of XP with COSMIC-FFP model. The core elements of GSOFT Practices; GSOFT Planning; GSOFT Estimation and GSOF Construction were adopted extensively, both by teachers and students throughout the study.

The results suggest that PP is an appropriate methodology for learning software programs. Both the case studies reveals that

- Making Stories gives better understanding of the assignments.
- Splitting the stories into task and finding the Cfsu helps the teacher to assign and evaluate the student's program consistently and accurately.
- Software selection for program development, based on the layer identification makes effective time management.

This study was checked by using the student volunteers and the strength was very meager to the study. The given assignments were real time project and took more person days to complete. The authors are interested in applying GSOFT in the short time program in a laboratory setting and as further study.

REFERENCES

[1] Ally M., Darroch F., and Toleman M.,2005. Framework for Understanding the Factors Influencing Pair Programming Success. *Proc. 6th Int'l Conf. Extreme Programming and Agile Process in Software Engineering*, Sheffield, UK, pp. 82-91.

[2] Beck K., and Andres C. 2003. *Extreme Programming Explained: Embrace Change,* 2nd Ed. Addison Wesley Professional.

[3] Braught G., Wahls T., and Marlin Eby. L. 2011. The Case for Pair Programming in the Computer Science Classroom. *ACM Transaction on Computing Education,* Vol. 1, Issue 11, pp. 2- 21.

[4] Cockburn A., and Williams L. 2001. The costs and benefits of pair programming, in Extreme programming examined. *Addison-Wesley Longman Publishing Co., Inc*. pp. 223-243.

[5] COSMIC – Common Software Measurement International Consortium 2007. The COSMIC Functional Size Measurement Method – version 3.0 Measurement Manual (The COSMIC Implementation Guide for ISO/IEC 19761:2003).

[6] Desharnais J.M. 2011. Using the COSMIC Method to Evaluate the Quality of the Documentation of Agile User Stories. *Joint Conference of the 21st International Workshop on Software Measurement*, pp. 269-272.

[7] Frent M. 2005. Correctness: A Very Important Quality Factor in Programming," *STUDIA UNIV. BABES, BOLYAI, INFORMATICA*, Vol. L, No. 1, pp. 11-20.

[8] Hannay J E., Tore D., Erik A., and Sjøberg D.I. k. 2009. The Effectiveness of Pair Programming : A Meta-Analysis. *Science Direct,* Vol. 51, Issue 7, pp. 1110-1122.

[9] Jeffrey C C., Lisa H., He L., Julia H., and Donna R. 2007. Increased Retention of Early Computer Science and Software Engineering Students using PP. *20th Conference on Software Engineering Education & Training (CSEET'07)* Dublin, Ireland, pp.115 – 122.

[10] Kanmani S., and Radhakrishnan P. 2010. A Hybrid Approach to Assess the Students Program Automatically," *Journal of the Indian Institute of Engineers India*, pp. 3-9.

[11] Miranda E. 2009. Sizing User Stories Using Paired Comparisons. *Information and Software Technology (IST) Journa*l, vol. 51, issue 9, pp. 1327-1337.

[12] Panagiotis S., Ioannis S., Lefteris A., and Ignatios D. 2009. An experimental investigation of the personality types impact on pair effectiveness in pair programming. *Empirical Software Engineering.* pp. 187–226.

[13] Radermacher A., Walia G., and Rummelt R. 2012. Assigning student programming pairs based on their mental model consistency: an initial investigation. *SIGCSE'12*, ACM. pp. 325-330.

[14] Ramli N., and Fauzi S.S.M. 2008. The Effects of PP in Programming Language Support. *International Symposium on Information Technology,* Vol. 1, pp. 1-4.

[15] Salleh N., Mendes E., Grundy J, and Giles J B. 2010. An Empirical Study of the Effects of Conscientiousness in Pair Programming Using the Five-Factor Personality Model. *ICSE'10,* Cape Town, South Africa. ACM, pp. 577-585.

[16] Tanja B T., Lepper A., and Schedding D. 2008. Pair Programming in software development teams- An empirical study of its benefits" *Information and Software Technology, Elsevier,* pp. 231-240.

[17] Walle T., and Hannay J E. 2009. Personality and Nature of Collaborations in Pair Programming. *Third International Symposium on Empirical Software Engineering and Management, IEEE*. pp. 203-213.

[18] Williams L.A., and Kessler R. 2000. The Effects of "Pair Pressure" and "Pair Learning" on Software Engineering Education. *Proceeding CSEET '00 Proceedings of the 13th Conference on Software Engineering Education & Training*. IEEE Computer Society Washington, pp. 59-65.

[19] www.extremeprogrammig.org.

[20] www.cosmicoa.com