ISSN (Print) : 0974-6846
Indian Journal of Science and Technology, *Vol 8(34), DOI: 10.17485/ijst/2015/v8i34/72194, December 2015* ISSN (Online) : 0974-5645

Correlation Study on Defect Density with Domain Expert Pair Speed for Effective Pair Programming

K. S. Sunitha* **and K. Nirmala**

Department of Computer Science, Quaid-E-Millath Government College for Women(A), University of Madras, Chennai - 600002, Tamil Nadu, India;
sunithasura2001@gmail.com, nimimca@yahoo.com

Abstract

Objectives: Proportional increase in speed of development (known as pair speed advantage) in software product development by pair programmers when compared with single programmer has been reported in literature. There is also an indication in the literature that the software defects in relation to lines of coding (known as defect density) are reduced in the case of pair programming when compared with conventional single programming technique. Correlation studies onthe contribution of application specific expert, when paired with conventional single programmer are not to be seen much in literature. Whether the findings seen in literature would hold good for small sized software developments also? Under these background this research work aims at presenting a correlation study (presentation of results) on the defect density of five chosen small sized software developments by three different programmer strategies, namely single, pair and expert programmer pair, in correlation with pair speed advantage. **Methods:** The research work however doesn't present correlation coefficients and other related statistical results, as 'correlation'as a term, is treated only for the study of relative performances. The novelty of the work is exhibited through the isolated study on the contribution of domain or application specific expert when acted as a pair alongside a relatively inexperienced s software programmer, while the domain expert need not know software programming as such, although he/she might still be a subject/domain specialist. **Finding:** The experimental works elaborated in the paper, which involved three s/w developer pairs on five small projects (file sizes varied from about 300 to 800 KB) were completely carried out by the control and direction of the researcher herself, in laboratory conditions as pointed out in the paper, and no external agencies were involved as the coding on the chosen applications is small when compared with huge LOC of s/w industries. These results were not sent to any other publisher for publication. Since the objective of the research work is to do a correlative study between the efforts of programmers and domain experts, MCA students and known domain experts (namely banking staff) were deployed by the researcher. The demography presented in the paper vouches the same. **Application/Improvement:** The paper has clearly demonstrated the performance improvement by the expert pair combination in the reduction of application specific defects (expressed in terms of defect density similar to the term pointed out by Frank Padberg et. al. 2003), when correlated with other two pairs. The paper also has shown that there is also a marginal increase in the pair speed advantage in the case of expert pair with compared with conventional programmer pair.

Keywords: Application Specific Experts, Pair Programming, Pair Speed Advantage, Software Defect Density

1. Introduction

Studies on defects in software under pair programming and on the development speed advantage of pair programming in isolation are found to be abundant in literature. But a correlation study of both is rare to be seen. One of the most important practices of Extreme Programming (XP) is pair programming[1] under agile technology. It is evident that pair programming due to the engagement of two programmers, would increase the

* *Author for correspondence*

personnel cost; but however aims at increasing the team productivity and also to improve the quality as compared to single programmer development[2]. On the other hand, 'it is debatable; whether or not the productivity gain is worth the extra cost' and 'it is also doubtful whether two programmers produce code twice as fast as a single programmer?'[3]. Pair programming have found to have provided only a small effect on the reduction of defects in large software development[4]. Software development in India is primarily manpower intensive and the growth is proportional to software engineer's employed[5] and this phenomenon has resulted in increase difficulty in coordinating and controlling large software developments. Small sized software developments are seen aplenty in South India. In view of the above background, this research work attempts to make a correlation (relation with each other) study on the defect density of five chosen small sized software development under three different programmer strategies under single, pair and expert programmer pair in line with pair speed advantage. The novelty of the work is exhibited on the isolated study of domain or application specific expert when acting as a pair alongside a relatively inexperienced software programmer, while the domain expert is not expected to know software programming as such, although he/she might be a subject/domain specialist. The objective of the research work is to segregate three types of defects such as syntactic, execution/running and domain or application specific and to make inferences on the role of expert's contribution in fixing the third type of defect. The work elaborated and presented in this research work is a part of a whole research program of the authors. Findings along with results reported in this research work would be of immense help to pair programming researchers and also to application specific software development teams in the field of software engineering.

2. Literature Support and Problem Formulation

Pair programming is one of the core areas of the process paradigm of XP[6]. At the expense of personnel costs increased by pairs, the productivity and the quality of software product is expected to increase, when compared with conventional single programmer's effort. Works on the quantification of pair programmer efforts in comparison with conventional single programmer for faster production has been reported. Advantages of pair programming and the specific issues arising out of it have been reported in detail by[7] Programming logics and coding output may be affected due to human factors and the programmers forming in pairs would influence the quality of the coding. Lessons are learnt from implementing the practices of project risk management during the process of software development[8] such as by pair programmers. This would indicate that the defect ratios of software projects would become sensitive. Pair speed advantage has been termed on the higher development speed by pair programmers than a single programmer[6]. The authors have also mathematically expressed the computation of software defect densities. Both these terms are restricted to software development costs and not on operation costs. According to the authors 'the average productivity of a single developer is measured in Lines of Coding (LOC) per month. This includes design, coding and unit testing, but excluding regression testing'. To reduce project cost, time of development, and to improve the customer expectations, XP coding technique has been suggested[9]. XP is one of the software development methodologies which are expected to improve the software quality particularly with pair programming. XP adopts agile technology to a great extent. This is intended to improve the productivity and also for adopting customer requirements. One of the elements of XP is pair programming that extensively reviews coding; an area of determining defects in software that is of demand now-a-days.

The importance of application specific errors in huge software development that involves several man years and the need for addressing such software bugs have been stressed[10]. Software developers generally pay more attention to commonly known errors caused by operating systems, but literatures have also reported on lesser known application dependent errors in software. Such violations of application specific coding rules are responsible for multitude of errors. Only application specific domain experts can discover such patterns of application dependent logical errors, so that they can get fixed relatively quickly. From the support of these base papers, the research work is identified to perform a correlation study on defect density through the contribution of domain specific experts in improving the pair speed advantage.

3. Proposed Method

3.1 Experimental Setup and Research Methodology

The proposed study has delimited with five small/medium sized software development products that are dealing with popular and less complex domain areas such as banking, hotel management etc. that are enlisted in this section. The pair group behavior, which is the main objective of the study, is planned with three categories (or cases) with a single programmer, pair of two programmers and most importantly a domain expert (in the chosen application area) alongside a programmer as a pair. The experiment is however focused only on the number of defects in the total LOC of each case and not focused on the algorithmic studies and efficiency of the coding. The experimental studies are limited to laboratory conditions and not on actual industrial environments. The demographic details with legends are presented below.

The software products are identified with P1, P2 etc. P1: Hotel management software; Single Programmer LOC: 2633; P2: Travel and Tours management software;(all Single Programmer) LOC: 3450; P3: Banking operations software; LOC: 4128; P4: Financial company management software; LOC: 7402; P5: Combined banking and financial borrowing/lending company software; LOC: 10239; The Programmer Cost (PC) is computed in terms of unit programmer cost which is computed as: One man day * fixed cost for one working day (assumed unit Rs) = 1. The PC can therefore be easily computed for actual cost based on any working day. The Expert Cost (EC) is determined for domain or application specific expert's unit cost for P1: 2.5 * PC; P2: 2.5* PC; P3: 2.7* PC; P4: 2.7* PC; P5: 2.7* PC. The factors, namely 2.5 or 2.7 can be changed with actual ratios practiced in real world condition. A well balanced demography has been chosen while developing the software in laboratory conditions. Experts acted as part of the control group samples but not concerned with the research program. All the samples of the control group for experiments are from the city of Chennai, India. Developments were carried out at various stages. As this research work forms a part of a whole research program, results have been shared for a few research programmes under the single supervisor/guide who is also the second author of this research work. Total programmers (sample) = 20. Demography of the programmers/experts is presented in Table 1.

4. Experimental Results and Discussion

Experiments were planned for development of software packages/products on five selective application areas by three programmer teams as specified earlier. At the development stage, before the testing process of the products, the number of defects caused in the coding is calculated along with the LOC. The demography details of the cases along with software product details are

Table 1. Demography of programmers/Experts for the experiments

ProjectId	Demography details																	
	Single Programmer				Pair Programmers								Expert Programmer Pair					
					Programmer 1			Programmer 2			Programmer			Expert				
	Age Limit	Programming Experience (yrs)	Position		Age Limit	Programming Experience (yrs)	Position		Age Limit	Programming Experience (yrs)	Position		Age Limit	Programming Experience (yrs)	Position		Age Limit	Domain Experience (yrs)	Position
P1	20	4	I to III year MCA students		21	4	I to III year MCA students		20	4	I to III year MCA students		20	4	I to III year MCA students		36	5	Professional consultants
P2	20	3			21	4			20	4			20	4			35	5	
P3	21	4			22	5			22	5			21	4			27	4	
P4	22	4			23	5			22	5			21	4			40	12	
P5	23	5			23	5			23	5			21	4			40	12	

presented in Table 1. As the research objective involves correlation study of these programmer groups, such a demographic detail is required.

The computation of pair speed advantage and the defect densities are presented below.

Development cost unit = Man days consumed * PC and/or EC.

Total single programmer man days: P1: 17; P2: 18; P3: 22; P4: 26; P5: 38.

Total pair programmers man days: P1: 28; P2: 28; P3: 30; P4: 32; P5: 44.

Total domain expert programmer pair man days: P1: 4*2.5 + 18; P2: 4*2.5 + 18; P3: 7*2.7 + 20;

P4: 6*2.7 + 24; P5: 10*2.7 + 26.

The defect density and pair speed advantage are computed as presented below[6]. Defect density: No. of defects lines per 1000 lines * (100% - good conventional coding that would have eliminated X% of these defects). Ex.: 400 / 1000 * (10/100) with a good coding that would have eliminated 90% of the defects (= 0.04).Pair speed advantage: Percentage of reduction of duration than single programmer duration. Ex.: For 17.65% shorter duration, the pair speed advantage would become 100 / (100 – 17.65) = 1.21.

A sample program coding is shown in Figure 1. The display (Figure 1) is just to indicate the vulnerability of defects liberally caused by inexperienced programmers and also allowed for the purpose of research. The fourth line in the Figure 1, namely the variable '*seatbook*' is erroneously spelt as 'seat-book'; fifth line shows '*flag=*' instead of specifying '*flag==*'. The eighth line of the coding specifies an application dependent value that is inadvertently buried in the program by the programmer that is causing a defect which was pointed out and rectified by the domain expert. In normal circumstances this rectification wouldn't have been possible.

```
void reserve (int n)
{if (n>arrRowState[14]) {cout<<"Too large group to accommodate"; getch (); return; }
int flag=0; int seat-book;
for (int i = 0 ; flag=0&&i<=13 ; i++) {
if (arr Row State [i] >= n) {flag=1;
cout<< "Following Seats Alloted";
seatbook = (((i)*5)+(6-arr Row State [i]));
for (int j = 0 ; j < n ; j++) {
cout<< ""<<seatbook+j<<"";
seat [(seatbook+j)]. is Empty=0; }
arr Row State [i] = arr Row State [i]-n;
arr Row State [14] = arr Row State [14]-n; } }
if (flag==0) {
while (n!=0) {
intmax, rowNo = 0; max=arr Row State [0];
for (int j = 0 ; j<14 ; j++) {
if (arr Row tate [j] > max) {
max = arr Row State [j];
row No = j; } }
if (n>max) {
n = n-max;
seatbook = (((row No)*5)+(6-arrRow State [rowNo]));
arrRowState [row No] = arr Row State [row No] - max;
for ( int j = 0 ; j<max ; j
++) {
cout<< ""<<(seatbook+j)<<"";
seat [(seatbook+j)]. is Empty=0; } }
else {
 reserve (n);  n=0; } } }
```

As stated earlier, the defects caused while coding have been grouped into three categories and the distribution of the ratios of individual category to the LOC of the three cases are presented in Table 2.

Table 2. Distribution of number of defects created by the three cases

Project Id	Ratio of Defects to LOC in Programs caused by								
	Single Programmer			Pair Programmers			Expert Programmer Pair		
	Syntactic Defects	Execution Defects	Application Specific Defects	Syntactic Defects	Execution Defects	Application Specific Defects	Syntactic Defects	Execution Defects	Application Specific Defects
P1	0.121	0.082	0.122	0.104	0.077	0.126	0.101	0.072	0.002
P2	0.118	0.098	0.080	0.101	0.066	0.100	0.098	0.063	0.003
P3	0.143	0.102	0.129	0.111	0.078	0.115	0.103	0.067	0.003
P4	0.138	0.047	0.100	0.100	0.023	0.100	0.094	0.024	0.001
P5	0.198	0.098	0.123	0.186	0.088	0.126	0.097	0.078	0.002

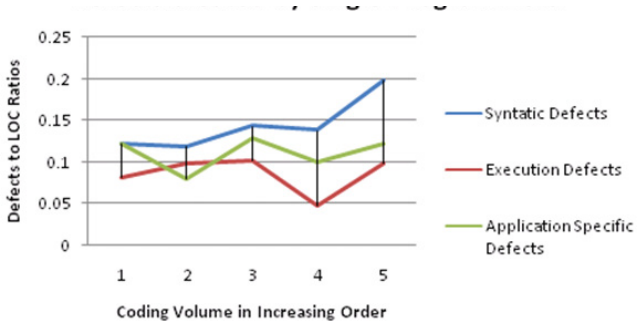

Figure 2. Distribution of defects to LOC ratios of single programmers.

The distribution presented in Table 2 is shown in Figures 2, 3 and 4 of each programmer category respectively. As seen from Figure 2 of the case of single programmer, the defect ratio of syntactic error supersedes other defects. Even though the distribution is shown in terms of increasing volume of coding, there is a dip in the case of 4th product (Figure 2). This may be due to the specific nature of relatively easier application. A similar observation is made in Figures3 and 4. It is quite evident that the ratio of application specific defects to LOC is the smallest in the case of expert programmer pair. This clearly indicates that the contribution of experts (application specific) in development is significant. There is a reduction of defects both syntactic as well as execution in the case of pair programmers as seen from Figure 3, when compared with Figure 2. This is in agreement with other published works.

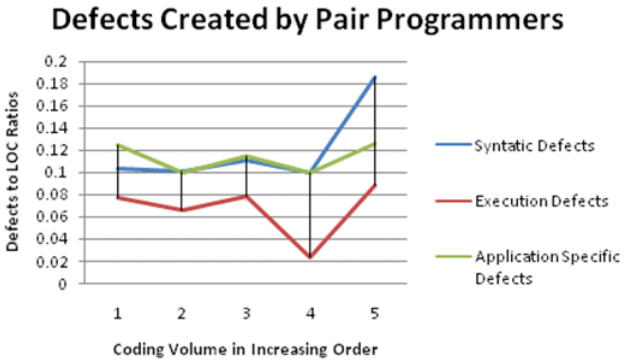

Figure 3. Distribution of defects to LOC ratios of pair programmers.

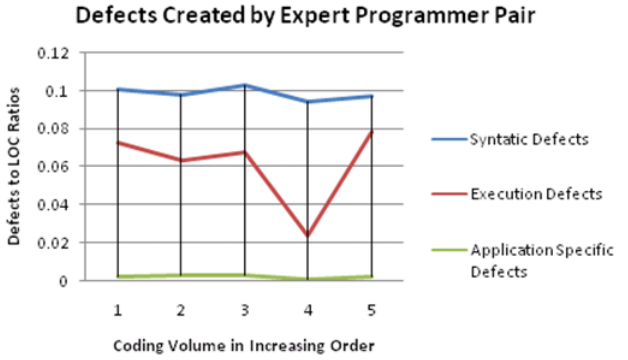

Figure 4. Distribution of defects to LOC ratios of expert programmer pairs.

The defect densities and speed advantages computed using the expressions explained earlier for the three groups of programmers are presented in Table 3.

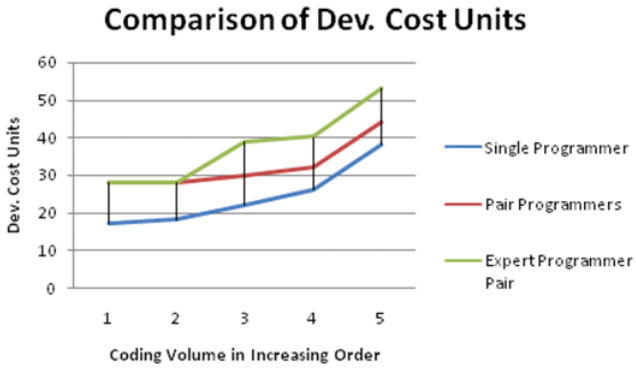

Figure 5. Distribution of Dev. cost units for the three cases.

Table 3. Distribution of defect densities and speed advantages of the three cases

	Single Programmer					Pair Programmers					Domain Expert Programmer Pair				
Project Id	S/w Product Size, KB	Dev. Cost units	Dev. Duration, days	Defect Density	Pair Speed Advantage	S/w Product Size, KB	Dev. Cost units	Dev. Duration, days	Defect Density	Pair Speed Advantage	S/w Product Size, KB	Dev. Cost units	Dev. Duration, days	Defect Density	Pair Speed Advantage
P1	332	17	17	0.04	1	367	28	14	0.032	1.21	380	28	18	0.01	0.95
P2	345	18	18	0.09	1	338	28	14	0.044	1.29	404	28	18	0.02	1.00
P3	394	22	22	0.12	1	403	30	15	0.09	1.47	487	38.9	20	0.02	1.10
P4	566	26	26	0.2	1	589	32	16	0.14	1.62	663	40.2	24	0.02	1.08
P5	867	38	38	0.34	1	822	44	22	0.22	1.73	892	53	26	0.03	1.46

The results are presented in Figures 5, 6 and 7 on the comparisons of costs, defect densities and pair speed advantage respectively.

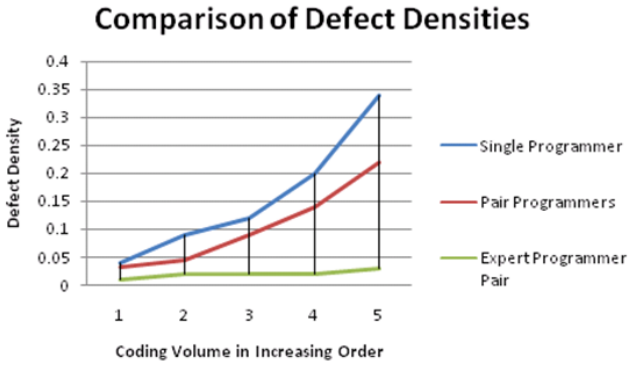

Figure 6. Distribution of defect densities for the three cases.

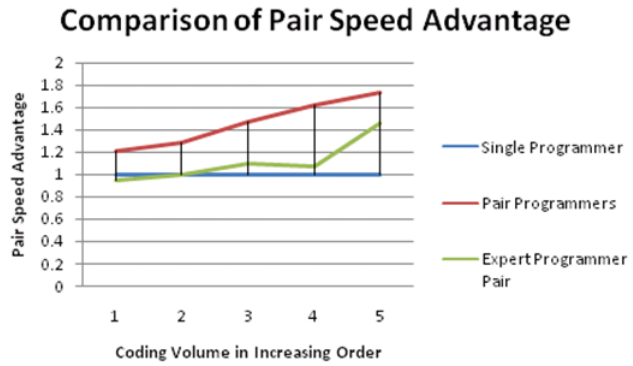

Figure 7. Distribution of pair speed advantages for the three cases.

The development unit costs of the three cases are gradually increasing along with the increase in volume of coding as seen from Figure 5. However the increase is seen rapid for the single programmer case while the gap is found to be gradually reducing in the case of pair programming from that of the case of single programming. But not so much seen in the case of expert programmer pair. This clearly indicates that the development cost might be reduced for large coding projects that are in agreement with other findings[3]. Important findings of this research demonstrate that the contribution of experts of the relevant domains would influence (reduce) the development cost for domain dependent developments.

Important observations are made from Figure 6. The defect densities are certainly found to be low in the case of expert programmer pair when compared with the rest of the two cases. Irrespective of the domain, the experts would certainly contribute in reducing the defects in the programs when paired with conventional programmers. Besides, the defect densities are found to be less in the case of pair programmers when compared with single programmer, as seen in Figure 6.

Unlike some of the other published works[6], there is no marked improvements seen in pair speed advantage as witnessed in Figure 7. However, large coding projects might show some improvements in pair speed advantage, as an indication is noted from project 1 to 5. There is an signal in Figure 7, that for larger coded projects, expert programmer pair might further increase the advantage of speed, as seen for this particular case in Figure 7. Yet again the advantage of pair speed is certainly visible in the case of pair programmers when compared with single programmer, which is in agreement with[6].

5. Conclusion

The experiments clearly indicate that there is a contribution by application specific experts in reducing the defect ratios, particularly on the application specific defects. There is also a marginal increase in the pair speed advantage in the case of expert pair with conventional programmer. Unlike the general belief that pair programmers would reduce production time when compared with single programmer, the reduction is found to be only very marginal in small sized software developments.

6. References

1. Huizinga D, Adam K. Automated defect prevention: Best practices in software management. John Wiley and Sons; 2007. p. 75.
2. Muller MM, Frank P. On the economic evaluation of XP projects. ACM SIGSOFT Software Engineering Notes. 2003; 28(5):168–77.
3. Kim N. Increasing quality with pair programming. Software Engineering [Master Thesis]. Thesis no: MSE-2003-15 06 2003.
4. Enrico diB, Ilenia F, Nattakaran P, Alberto S, Giancario S, Jelena V. Pair Programming and Software Defects – A Large, Industrial Case Study. IEEE Transactions on Software Engineering. 2013; 39(7)930–53.
5. Rishikesha T, Krishnan NP, Ganesh NP. Software product development in lesson: Lessons from Six Cases. Journal of India in the Global Software Industry: Innovation, Firm Strategies and Development; 2003. p. 139–63.

6. Padberg F, Matthias MM. Analyzing the cost and benefit of pair programming. Proceedings Ninth International in Software Metrics Symposium, 2003. IEEE; USA; 2003. p. 166–77.
7. Williams L. Integrating pair programming into a software development process. Proceedings 14th Conference on in Software Engineering Education and Training, 2001 IEEE; 2001. p. 27–36.
8. Kwak YH, Jared S. Project risk management: Lessons learned from software development environment. Technovation. 2004; 24(11):915–20.
9. Kent B. Extreme Programming Explained: Embrace Change. 2nd Edition. Addison-Wesley; NY: USA; 2004.
10. Livshits B, Thomas Z. DynaMine: Finding common error patterns by mining software revision histories. In ACM SIGSOFT Software Engineering Notes. ACM. 2005; 30(5):296–305.
11. Nagarajan R, Velanganni JA, Sujatha S. Behavioural Aspects of Software Project Management-In-House Software Development. Indian Journal of Science and Technology. 2015; 8(S3):1–9.