Simulator for Software Project Inspection

P. K. Suri[1], Bharat Bhushan[2], Ashish Jolly[3]

[1]*Department of Computer Science & Applications, Kurukshetra University, Kurukshetra (Haryana), India.*
[2]*Department of Computer Science & Applications, Guru Nanak Khalsa College, Yamuna Nagar (Haryana), India.*
[3]*Department of Computer Science & Applications, Shri Atmanand Jain Institute of Management & Technology, Ambala City (Haryana), India.*

Abstract
Software project inspection has been shown to be an effective defect removal practice, leading to higher quality software with lower field failures. The use of software code inspections, design inspections, and requirements inspections, has been found to increase software quality and lower software development costs [1, 2].
Efficiency is the main attribute of reliability. Efficiency measures the performance of the software and performance of software is better if it is error free or defect free. To check the defect free software and to make it acceptable in the market, the software is inspected by the analysts on various criteria. The criteria are termed as defects classification and they are described as defects like Logical, User Interface, Design Issues, Hard Coding, Modularity etc.
An attempt has been made to design a simulator to inspect the software on the basis of certain criteria. The software is divided into ten modules and each module is inspected by fourteen analysts. Each analyst gives his view about the different criteria. The rate of agreement among the analyst is computed on the basis of Fleiss Kappa Coefficient using various relations. The value of Kappa Coefficient decides whether the analysts are agreed on these criteria. If they agree on these criteria, then the software is treated to be better and more efficient as compared to the previous version.

Keywords
Defect analysis, Defect Classification, Efficiency, Fleiss Kappa Coefficient, Inspection

1. Introduction

Inspections are defined as a static analysis technique that relies on visual examination of development products to detect errors, violations of development standards, and other problems [3]. It also helps to increase the development team's familiarity with the code. Prior studies indicate that software project inspections
can detect as little as 20% to as much as 93% of the total number of defects in an artifact. Based upon a literature survey report, on average software inspections find 57% of the defects in code and design documents. Inspections have been traditionally done manually with key members of the development and quality assurance teams. [4]

Industrial data has shown that inspections are among the most effective of all verification and validation (V&V) activities, measured by the percentage of defects typically removed from a document. [5]

Manuscript received August 5, 2009
Manuscript revised August 20, 2009

In many software organizations, defects are classified very simply, using categories such as Minor, Major, Severe, Critical. Simple classifications of this kind are typically used to assign priorities in repairing defects. Deeper understanding of the effectiveness of software development methodologies and techniques require more detailed classification of defects. [6]

Although no one is happy to find defects in their software, defects are introduced and removed continually during software engineering processes, and it is practically necessary to acknowledge, record, and analyze those defects to make progress toward higher standards of quality. [7]

Most software reliability methods have been developed to predict the reliability of a program using only data gathered during the testing and validation of a specific program. Hence, the confidence that can be attained in the reliability estimate is limited since practical resource constraints can result in a statistically small sample set. [8]

Fleiss' Kappa
Fleiss' kappa is a variant of Cohen's kappa, a statistical measure of inter-rater reliability. Where Cohen's kappa works for only two raters, Fleiss' kappa works for any constant number of raters giving categorical ratings to a fixed number of items. It is a measure of the degree of agreement that can be expected above chance. Agreement can be thought of as follows, if a fixed number of analysts assign numerical ratings to a number of efficiency measure criterions then the kappa will give a measure for how consistent the ratings are. The kappa, κ, can be defined as

$$\kappa = \frac{PAV - PE}{1 - PE}$$

The factor 1 - PE gives the degree of agreement that is attainable above chance, and PAV - PE gives the degree of agreement actually achieved above chance. The statistic takes values between 0 and 1, where a value of 1 means complete agreement.

With appropriate interpretation, the Kappa value can be used as an objective criterion for evaluating the quality of the software.

Kappa Statistic	Strength of Agreement
< 0	Poor agreement
0.0 – 0.20	Slight agreement
0.21 – 0.40	Fair agreement
0.41 – 0.60	Moderate agreement
0.61 – 0.80	Substantial agreement
0.81 – 1.00	Almost perfect agreement

Table 1: Levels of Agreement among Analysts

2. Proposed Simulator

Software inspection plays a crucial role in achieving high quality software right from the beginning. Especially for requirements documents inspections are beneficial as defects can be detected and removed at an early point in time before they can leak into subsequent phases of the development process, where those defects can cause high rework cost and quality problems [9].

The Proposed Simulator for software acceptance assumes that whether the new version of the software to be released in the market will be acceptable or not based on certain criteria. If the efficiency of the new software will be higher than the previous versions, the new version of the software will be more efficient. Efficiency is one of the major attribute of software quality. The software is inspected on the basis of various criteria and it is checked whether the analysts are agreed upon these criteria. The simulator is designed to determine the efficiency level of agreement among many analysts which is measured using Fleiss Kappa coefficient. The software under study consists of N number of modules. There are k numbers of criteria. A large number of analysts are given the responsibility to inspect the various modules of the software and using Fleiss Kappa Coefficient it is determined at what level all the analysts are agreed upon to launch the new version of the existing software. The efficiency level of the software will be an indicator for the quality of the new version of the software.

Assumptions

1. There are fixed number of analysts.
2. The number of analysts select the criteria is computed with the help of random number generator program.
3. Software consists of a large number of modules.

Terms and Notations

N	: Number of different software modules.
A	: Number of Software Analyst.
k	: Various types of criteria
PE	: Expected Probability of disagreement
P1(J)	: Proportion of all efficiency levels which were to the J-th criteria.
P(I)	: The extent to which analysts agree for the I-th module.
PAV	: Mean of PI’s
A[I,J]	: Efficiency levels for each module of the software using random number generator program.
κ	: Fleiss Kappa Coefficient

Algorithm:

Step 1: Start

Step 2: Read N, A, k

Step 3: Compute the efficiency levels for each module of the software using random number generator program through Poisson Distribution, (A[I,J]).

Step 4: Compute sum of all criteria values on the basis of which the software is inspected.

Step 5: Compute proportion of all efficiency levels which were to the J-th category using the relation

$$P1(J) = \frac{1}{NA}\sum_{i=1}^{N} A_{ij}$$

Step 6: Compute P(I), the extent to which analyst agrees for the I-th module using the relation

$$P(I) = \frac{1}{A(A-1)}\left[\left(\sum_{j=1}^{k} A_{ij}^{2}\right) - (A)\right]$$

Step 7: Compute PAV i.e., the mean of PI’s using the relation

$$PAV = \frac{1}{N}\sum_{i=1}^{N} P(I)$$

$$= \frac{1}{NA(A-1)}\left(\sum_{i=1}^{N}\sum_{j=1}^{k} A_{ij}^{2} - NA\right)$$

Step 8: Compute PE i.e., the expected probability of disagreement using the relation

$$PE = \sum_{j=1}^{k} P1(J)^2$$

Step 9: Compute Fleiss' Kappa coefficient using the relation

$$\kappa = \frac{PAV - PE}{1 - PE}$$

Step 10: Stop.

3. Case Study:

Fourteen analysts (A) are assigned the job of software inspection and they study the software modules on the basis of certain criteria among various modules of the software. For this single software is recommended and inspected for certain type of criteria (k) based on ten different modules (N) of the same software.

Case 1

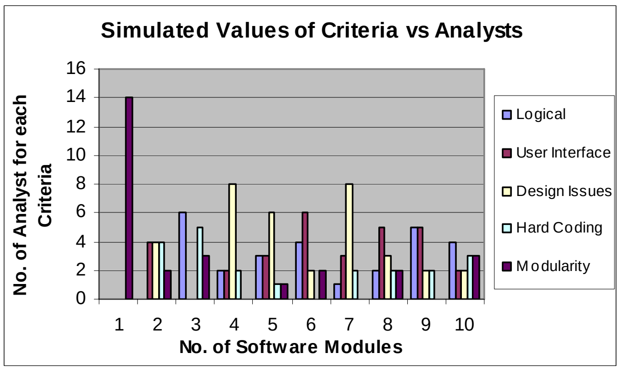
Ten modules of the software are inspected on the basis of five types of software criteria. The criteria are Logical, User Interface, Design Issues, Hard Coding and Modularity designated as 1, 2, 3, 4 and 5 as shown in table 2. P(I) and P1(J) are computed using the relations defined above.

Input: Read the value of Number of different software modules (N), Number of software analysts (A) and various types of criteria (k).

Table 2 shows the simulated values of criteria vs software modules and is depicted in graph 1.

Criteria → / S/w Modules ↓	1	2	3	4	5
1	0	0	0	0	14
2	0	4	4	4	2
3	6	0	0	5	3
4	2	2	8	2	0
5	3	3	6	1	1
6	4	6	2	0	2
7	1	3	8	2	0
8	2	5	3	2	2
9	5	5	2	2	0
10	4	2	2	3	3

Table 2



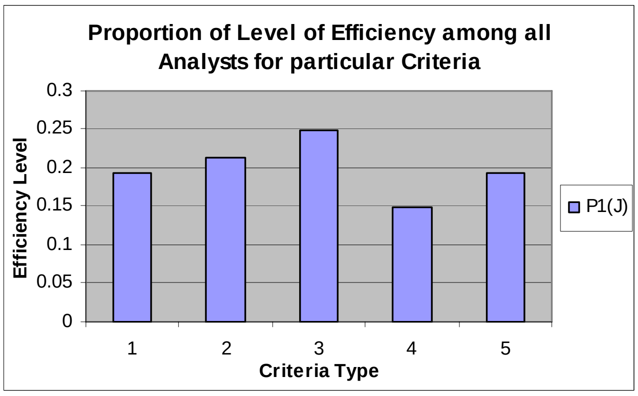
Graph 1

Output:

i) Table 3 shows the proportion of all efficiency levels which were to the J-th criteria and is depicted in graph 2.

Criteria	1	2	3	4	5
P1(J)	0.1929	0.2143	0.2500	0.1500	0.1929

Table 3

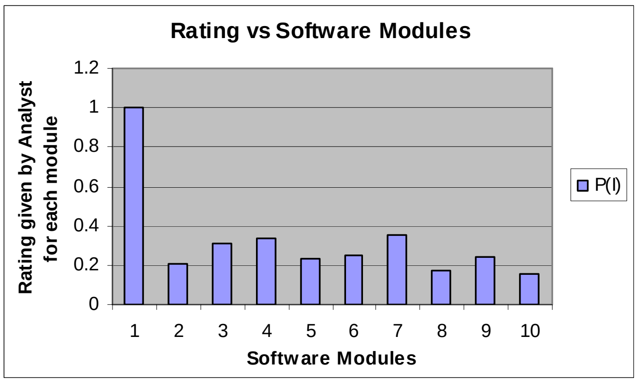


Graph 2

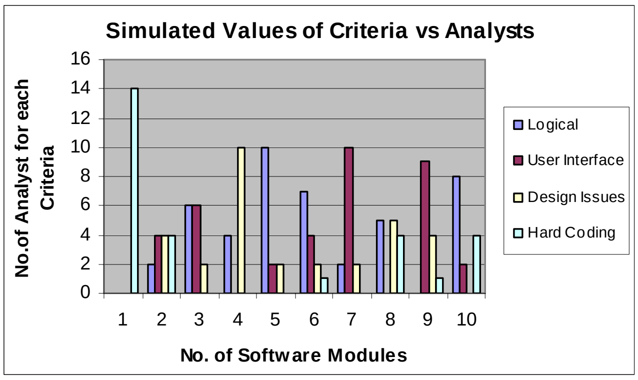
ii) Table 4 shows the extent to which analysts agree for the I-th module and is depicted in graph 3.

S/w Modules	P(I)
1	1.0000
2	0.2088
3	0.3077
4	0.3407
5	0.2308
6	0.2527
7	0.3516
8	0.1758
9	0.2418
10	0.1538

Table 4



Graph 3



Graph 4

iii) Fleiss Kappa Coefficient

Mean of PI's = PAV = 0.3264
Expected Probability of disagreement,
PE = 0.2053
Kappa Coefficient, κ = 0.1523

Case 2

Ten modules of the software are inspected on the basis of four types of software criteria. The criteria are Logical, User Interface, Design Issues and Hard Coding designated as 1, 2, 3 and 4 depicted in table 5. P(I) and P1(J) are computed using the relations defined above.

Input: Read the value of Number of different software modules (N), Number of software analysts (A) and various types of criteria (k).
Table 5 shows the simulated values of criteria vs software modules and is depicted in graph 4.

Criteria → / S/w Modules ↓	1	2	3	4
1	0	0	0	14
2	2	4	4	4
3	6	6	2	0
4	4	0	10	0
5	10	2	2	0
6	7	4	2	1
7	2	10	2	0
8	5	0	5	4
9	0	9	4	1
10	8	2	0	4

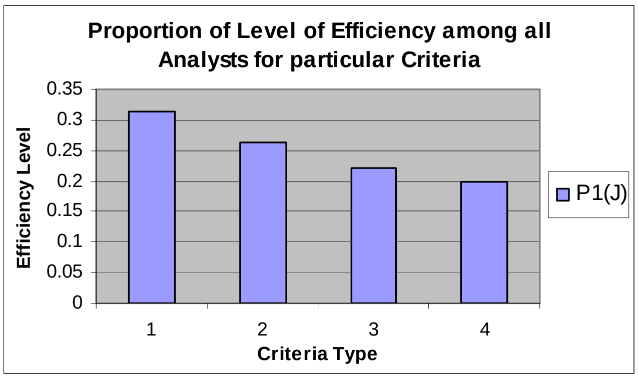
Table 5

Output:

i) Table 6 shows the proportion of all efficiency levels which were to the J-th criteria and is depicted in graph 5.

Criteria	1	2	3	4
P1(J)	0.3143	0.2643	0.2214	0.2000

Table 6

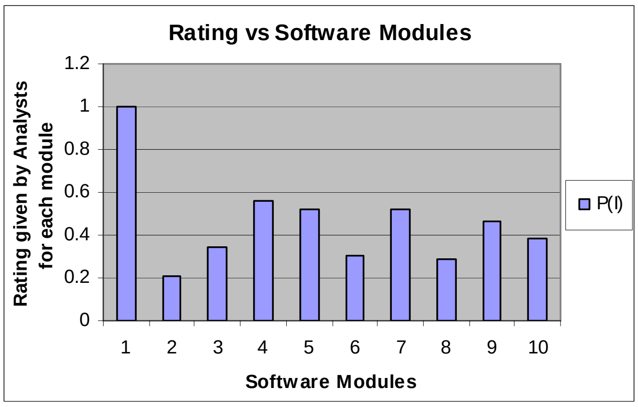


Graph 5

ii) Table 7 shows the extent to which analysts agree for the I-th module and is depicted in graph 6.

S/w Modules	P(I)
1	1.0000
2	0.2088
3	0.3407
4	0.5604
5	0.5165
6	0.3077
7	0.5165
8	0.2857
9	0.4615
10	0.3846

Table 7



Graph 6

iii) Fleiss Kappa Coefficient

Mean of PI's = PAV =0.4582
Expected Probability of disagreement,
PE = 0.2577
Kappa Coefficient, κ = 0.2702

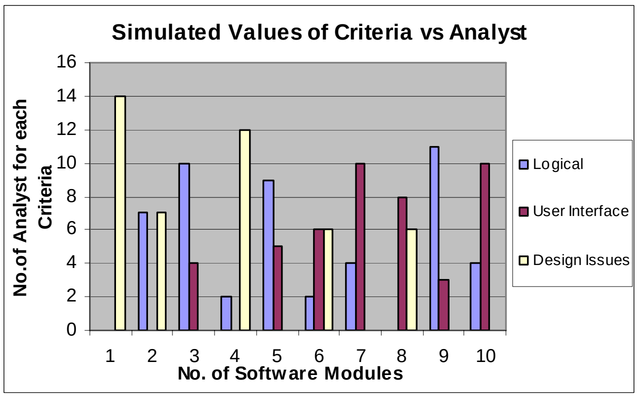
Case 3

Ten modules of the software are inspected on the basis of three types of software criteria. The criteria are Logical, User Interface and Design Issues designated as 1, 2 and 3 depicted in table 8. P(I) and P1(J) are computed using the relations defined above.

Input: Read the value of Number of different software modules (N), Number of software analysts (A) and various types of criteria (k).
Table 8 shows the simulated values of criteria vs software modules and is depicted in graph 7.

Table 8

Criteria ⇨ ⇩ S/w Modules	**1**	**2**	**3**
1	0	0	14
2	7	0	7
3	10	4	0
4	2	0	12
5	9	5	0
6	2	6	6
7	4	10	0
8	0	8	6
9	11	3	0
10	4	10	0



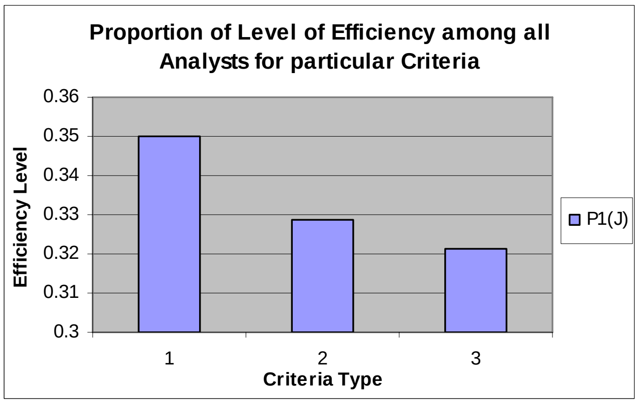
Graph 7

Output:

i) Table 9 shows the proportion of all efficiency levels which were to the J-th criteria and is depicted in graph 8.

Criteria	**1**	**2**	**3**
P1(J)	0.3500	0.3286	0.3214

Table 9

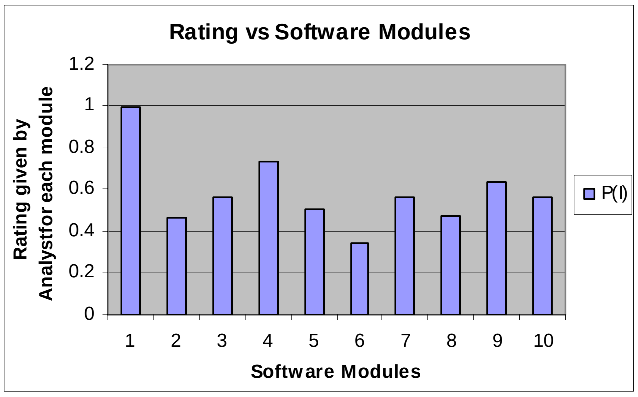


Graph 8

ii) Table 10 shows the extent to which analysts agree for the I-th module and is depicted in graph 9.

S/w Modules	**P(I)**
1	1.0000
2	0.4615
3	0.5604
4	0.7363
5	0.5055
6	0.3407
7	0.5604
8	0.4725
9	0.6374
10	0.5604

Table 10



Graph 9

iii) Fleiss Kappa Coefficient

Mean of PI's = PAV =0.5835

Expected Probability of disagreement,
PE =0.3338

Kappa Coefficient, κ = 0.3749

4. Discussion and Conclusion

The present simulator is designed to compute the level of agreement among different analysts. This level of agreement is described the level of efficiency which is computed with the help of Fleiss Kappa Coefficient.

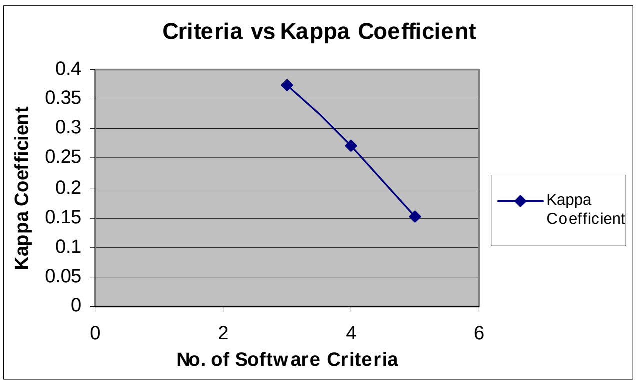
Software Inspection is an important part of the software project management. To launch the new version of the software in the market, there must be some level of agreement among analysts. The software is inspected on important criteria for which the old version of the software has some discrepancies. The newer version is designed in such a manner that the discrepancies of the older version are removed. Therefore a team of analysts are given the job of software inspection on various criteria. The level of agreement among the analysts is measured using kappa coefficient. It is found that if the software is inspected on fewer numbers of criteria then the level of agreement will be higher and the new version of the software will fulfill the criteria more in comparison with the older version.
In other words, the newer version will be more efficient and software will be released without any discrepancy in the market.

Table 11 shows the value of kappa coefficient vs number of criteria as discussed above in case 1, case 2 and case 3.

Table 11

No. of Criteria	**Kappa Coefficient**	**Strength of Agreement**
5	0.1523	Slight agreement
4	0.2702	Fair agreement
3	0.3749	Fair agreement

Our proposed simulator results satisfy the above description as depicted in the graph 10 as shown below.



Graph 10

References

[1] Fagan, M. E., "Design and Code Inspections to Reduce Errors in Program Development," *IBM Systems Journal*, vol. 15, pp. 182-211, 1976.

[2] O'Neill, D., "Issues in Software Inspection," *IEEE Software*, 1997, pp. 18-19.

[3] IEEE, "IEEE 1028-1988: IEEE Standard for Software Reviews and Audits," 1988.

[4] Nachiappan Nagappan, Laurie Williams, Mladen Vouk, "Preliminary Results on Using Static Analysis Tools for Software Inspection", Department of Computer Science, North Carolina State University, Raleigh, NC, USA.

[5] Rus, I., Shull, F., Donzelli, P., "Decision Support for Using Software Inspections," presented at 28th Annual NASA Goddard Software Engineering Workshop, 2003. pp. 3 -11

[6] Diane Kelly, Terry Shepard, "A Case Study in the Use of Defect Classification in Inspections", IBM Centre for Advanced Studies Conference, Toronto, Ontario, Canada pp. 7, 2001.

[7] Siddhartha Dalal, Michael Hamada, Paul Matthews, Gardner Patton, "Using Defect Patterns to Uncover Opportunities for Improvement", Bellcore Morristown, New Jersey.

[8] Raymond A. Paul, Farokh Bastani, I-Ling Yen, Venkata U.B. Challagulla, "Defect-Based Reliability Analysis for Mission-Critical Software," Compsac, pp.439, The Twenty-Fourth Annual International Computer Software and Applications Conference, 2000.

[9] Barry Boehm, Vic Basili, "Defect Reduction Top 10", IEEE Software, January 2001.

Dr. P.K. Suri received his Ph.D degree from Faculty of Engineering, Kurukshetra University, Kurukshetra, India and master's degree from Indian Institute of Technology, Roorkee (formerly known as Roorkee University), India. He is working as Professor in the Department of Computer Science & Applications, Kurukshetra University, Kurukshetra - 136119 (Haryana), India since Oct. 1993. He has earlier worked as Reader, Computer Sc. & Applications, at Bhopal University, Bhopal from 1985-90. He has supervised ten Ph.D.'s in Computer Science and eleven students are working under his supervision. He has more than 100 publications in International / National Journals and Conferences. He is recipient of 'THE GEORGE OOMAN MEMORIAL PRIZE' for the year 1991-92 and a RESEARCH AWARD –"The Certificate of Merit – 2000" for the paper entitled ESMD – An Expert System for Medical Diagnosis from INSTITUTION OF ENGINEERS, INDIA. His teaching and research activities include Simulation and Modeling, SQA, Software Reliability, Software testing & Software Engineering processes , Temporal Databases, Ad hoc Networks, Grid Computing , and Biomechanics.

Dr. Bharat Bhushan received his Ph.D degree from Department of Computer Science & Applications, Kurukshetra University, Kurukshetra, M.Sc (Physics) from Punjab University Chandigarh and M.Sc (Comp. Sc.), MCA degrees from Guru Jambeshwar University, Hissar in 2001 respectively. Presently working as Head, Department of Computer Science and Applications, Guru Nanak Khalsa College, Yamuna Nagar (affiliated to Kurukshetra University, Kurukshetra- Haryana, India) and senior most teacher of computer science in Haryana since 1984. He is a member of Board of Studies of Computer Science, Kurukshetra University. His research interest includes Software Engineering, Digital Electronics and Simulation Experiments.

Ashish Jolly received his MCA degree from University of Madras, Chennai in the year 1999. Currently he is pursuing Ph.D in Computer Science from Department of Computer Science & Applications, Kurukshetra University, Kurukshetra, India. He is working as a Asstt Professor and Head in the Department of Computer Science & Applications, Shri Atmanand Jain Institute of Management & Technology (affiliated to Kurukshetra University, Kurukshetra), Ambala City, Haryana, India. His research area includes Simulation, Software Engineering and Software Project Management.