Global and Stochastic Analysis
Vol. 4 No. 1, January (2017), 119-125

MUK PUBLICATIONS
Open Access Publisher

IMPORTANCE OF ENVIRONMENTAL FACTORS AFFECTING SOFTWARE RELIABILITY

A. LOGANATHAN AND R. JEROMIA MUTHURAJ*

Abstract. This paper presents the findings of empirical results from 25 participants from system engineers, programmers, testers, software development managers, and other people involved in software development. In 2000 Xuemei Zhang, Hoang Pham described Thirty-two factors which are affecting the software reliability. In this study another 2 factors are added which are affecting the software reliability and the study identifies the factors which have significant impact on software reliability.Relative weight, Correlation analysis and principal component analysis are used to identify the significant factors. The findings may have important implications for further research and it may be useful for software development.

1. Introduction

Software plays an important role all over the world. All human beings are using software in their daily life. So the importance of software is very high. And the reliability of software is also unavoidable. Software Reliability is defined as the probability of failure-free software operation for a specified period of time in a specified environment. Software reliability evaluation is playing an important role in software reliability engineering. The role of statistics is also very important in reliability estimation for software. There are many hardware reliability approaches but Software Reliability Modeling (SRM) work started in the early '70s with the inventive works of Jelinski and Moranda, Shooman and Coutinho. After that many works were done related to software reliability. Many software reliability models were constructed in parametric and non-parametric approaches. Some parametric models are Jelinski and Moranda De-Eutrophication Model (1972), Schick and Wolver ton Model, Goel and Okumoto Imperfect Debugging Model, Littlewood - Verrall Bayesian Model (1973), Goel-Okumoto Nonhomogeneous Poisson Process Model, Shooman Exponential Model, and etc. Some Non Parametric models are A Non-Parametric Order Statistics Software Reliability Model (1998), State Transition Model for Predicting Software Reliability (2007), and etc. The experts say that there are more than 225 software reliability models. But there is not even a single model that can be used in all situations. A model may work well for a set of certain software, but it may be completely off track for other kinds of problems. In 2000 Xuemei Zhang, Hoang Pham introduced a new way of estimating

2000 *Mathematics Subject Classification.* Primary 62N05; Secondary 62H25.
Key words and phrases. Software reliability, Relative weight method, Principal Component Analysis, Kendalls correlation .
* Corresponding Author.

software reliability with some environmental factors which are affecting the software. They have defined 32 environmental factors and they conducted a study to find the significant factors which are affecting the reliability of the software. The questioner has 32 variables with 8 scale measurements that 0 to 7. If the measure is 0 then the factor is not significant and if the measurement is 7 the factor is highly significant. They received the information from 23 software development practitioners from various software development companies. They use some statistical tools Relative weight method, Factor analysis, Correlation analysis and ANOVA. Finally the significant factors are discussed. In 2015 Mengmeng Zhu, Xuemei Zhang, and Hoang Pham revisit the 32 environmental factors and analyze their impact on software development and reliability based on a current survey to software development practitioners. In this study, the questioner is formed and performed a survey to get a quantitative and qualitative data from managers, software engineers, designers, programmers and testers who participate in the software development practice. Thirty four factors are involved in every phase of the software development process and the information about the background of survey participants are considered in this research. The Investigative analyses are used to identify the most significant factors with respect to software reliability and study the correlation of these factors. Statistical tools such as relative weighted method, Kendalls correlation analysis are applied to analyze these factors.

2. The objectives of this paper

The main objective of this work is to reinvestigate the factors influencing change. Software reliability environmental factors are proposed before two decades (2000 by Xuemei Zhang, Hoang Pham). The usage of internet is high now and the usage of the software is also more. Secondary objective is the buyer (customers) convenience how to get good software? Is a question. This study may give answer for that question.

3. Statistical methodologies

This study utilized several methods to analyze the ranking data. Relative weight method, Principal component analysis (PCA) and Kendall's correlation analysis.

3.1. Relative weight method. The relative weight method was used to obtain the final ranking for the factors. Let r_{ij}be the original ranking of the i$_{th}$ factor on the j$_{th}$ survey. As discussed in 2000(H. Pham). First normalize these r$_{ij}$such that

$$w_{ij} = \frac{r_{ij}}{\sum_{i=1}^{n} r_{ij}}$$

Where n is the number of factors on the j$_{th}$ survey

Therefore $\sum_{i=1}^{n} w_{ij} = 1$ for all j. Then average these w$_{ijis}$ to obtain the final weight for the i$_{th}$ factor such that

$$w_i^* = \frac{\sum_{j=1}^{l} w_{ij}}{l}$$

Where l is the number of surveys used in this method. Based on these relative weights, we obtain the final weight for each factor.

3.2. Kendall's correlation analysis. Kendall's Tau correlation coefficient is calculated from a sample of N data pairs (X, Y) a variable U as the ranks of X and a variable V as the ranks of Y. Kendall's Tau is then calculated from U and V using

$$\hat{\tau} = \frac{2(n_C - n_D)}{\sqrt{N(N-1) - T_X} - \sqrt{N(N-1) - T_Y}}$$

$$T_X = \sum_{i=1}^{S_X} (t_{(X)i}^2 - t_{(X)i})$$

$$T_Y = \sum_{i=1}^{S_Y} (t_{(Y)i}^2 - t_{(Y)i})$$

The parameter n_C is the total number of concordant pairs and n_D is the total number of discordant pairs.

3.3. Principal component analysis. The goals of PCA are to

- Extract the most important information from the data table.
- Compress the size of the data set by keeping only this important information.
- Simplify the description of the data set.
- Analyze the structure of the observations and the variables.

4. Data description

In this study the questioner is formatted with 34 variables andthe factors which is affecting the software reliability. The respondents have to respondfor each variable with the scale measurement 0 to 7. The participants are software development managers, system engineers, programmers, testers and administrators. Totally 25 numbers of participants are responded.

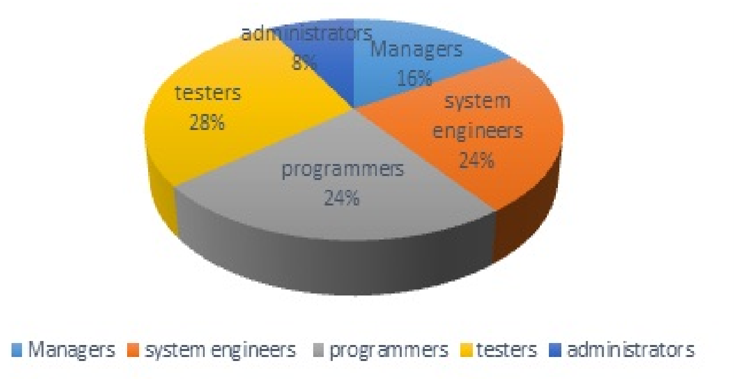

FIGURE 1. Data description

5. Finding & Results

5.1. Relative Weight Method Ranking. Results by the relative weight method are given in Table 1. The column named Normalized Priorities gives the contribution of each factor. For example, Programmer skill contributes approximately 4.06% (its relative weight 0.040609). Higher priority value indicates higher ranking. Since lower class rank implies decrease in magnitude of relative importance, software developers should then pay more attention to the factors with high ranks. Then final priority information can be used to guide the software development process of different applications.The table gives the list of all the environmental factors which are important for software development process and to work with the software. According to The scores which are given by the experts the significant factors are arranged in ascending order.

TABLE 1. Relative Weight Method Ranking

Rank	Factor Number	Factor name	Normalized priorities
1	F16	Programmer skill	0.040609
2	F8	Frequency of program specification change	0.040552
3	F23	Testing effort	0.039559
4	F22	Testing environment	0.038552
5	F1	Program complexity	0.038076
6	F25	Testing methodologies	0.037368
7	F6	Percentage of reused modules	0.036619
8	F26	Testing coverage	0.036325
9	F27	Testing tools	0.035769
10	F11	Requirements analysis	0.035574
11	F17	Programmer organization	0.034895
12	F24	Testing resource allocation	0.033364
13	F4	Amount of programming effort	0.032157
14	F10	Design methodology	0.031899
15	F20	Domain knowledge	0.031685
16	F13	Programmers experience	0.031536
17	F7	Programming language	0.031156
18	F3	Difficulty of programming	0.031105
19	F5	Level of programming technologies	0.030483
20	F9	Volume of program design documents	0.030157
21	F2	Program categories	0.029964
22	F18	Development team size	0.029749
23	F15	Development management	0.02923
24	F12	Relationship of detailed design to requirement	0.027203
25	F14	Work standards	0.024024
26	F19	Program workload (stress)	0.023285
27	F33	System software	0.022551
28	F28	Documentation	0.021772
29	F34	Random access memory	0.021184
30	F21	Human nature	0.02004
31	F29	Processors	0.014883
32	F30	Storage devices	0.013609
33	F31	Input/output devices	0.012697
34	F32	Telecommunication devices	0.012369

5.2. Kendall's correlation analysis. The purpose of performing correlation analysis is to observe the relation between variables and find out the strength and direction of this relationship. Having the knowledge of the correlation of the environmental factors (table 2)will provide a better understanding for software developers on resource allocation and testing efficiency during the software development and testing. Here in this study Kendall s correlation is used to see the correlated factors. Table 2 gives the result how the factors are correlated with other factors.

TABLE 2. correlation analysis

Factor Number	Factors	Correlated factors	Correlation
1	Program complexity	Programmer skill	0.519
		Design methodology	0.489
2	Program categories	Difficulty of programming	0.591
3	Difficulty of programming	Program categories	0.591
		Programming language	0.466
		Work standards	0.509
4	Amount of programming effort	Percentage of reused modules	0.459
		Program workload	0.438
5	Level of programming technologies	Frequency of program specification change	0.699
		Volume of program design documents	0.59
6	Percentage of reused modules	Amount of programming effort	0.459
7	Programming language	Difficulty of programming	0.466
8	Frequency of program specification change	Level of programming technologies	0.699
9	Volume of program design documents	Level of programming technologies	0.59
10	Design methodology	Program complexity	0.489
		Testing environment	0.754
11	Requirements analysis	Domain knowledge	0.587
12	Relationship of detailed design to requirement	Testing resource allocation	0.64
		Work standards	0.541
13	Programmers Experience	Programmer organization	0.782
		Programmer skill	0.654
14	Work standards	Difficulty of programming	0.509
		Relationship of detailed design to requirement	0.541
15	Development management	Testing methodologies	0.552
16	Programmer skill	Program complexity	0.519
		Programmers Experience	0.654
17	Programmer organization	Programmers Experience	0.782
		Program workload	0.456
18	Development team size	Testing coverage	0.523
19	Program workload	Programmer organization	0.456
		Amount of programming effort	0.438
20	Domain knowledge	Requirements analysis	0.587
21	Human nature	Processors	0.65
		Input/output devices	0.458
		Documentation	0.311
22	Testing environment	Design methodology	0.754
23	Testing effort	Testing coverage	0.654
24	Testing resource allocation	Relationship of detailed design to requirement	0.64
25	Testing methodologies	Development management	0.552
		Testing tools	0.511
26	Testing coverage	Development team size	0.523
		Testing effort	0.654
27	Testing tools	Testing methodologies	0.511
28	Documentation	Human nature	0.311
29	Processors	Human nature	0.65
30	Storage devices	Telecommunication devices	0.544
31	Input/output devices	Human nature	0.458
32	Telecommunication devices	Storage devices	0.544

5.3. Principal Component Analysis. Principle component analysis (PCA) is a statistical analysis that can be used to reduce the dimensionality of a data set consisting of a large number of interrelated variables. The idea is that we can use smaller dimension set of principle components to capture the characteristics of the larger data set and provide a concise yet critical principle components for software developers. In this study for all the 34 factors PCA procedure is applied. 100% of variation is explained by 34 factors. The PCA is explaining how the variation is explained. If we convinced with 65% of variation then we can take only top 6

TABLE 3. PCA Result

Component	Total	Variance	Cumulative % of Variance	F1	F2	F3	F4	F5	F6	F7
1	6.869	20.202	20.202	-.067	.356	.692	.281	.124	.473	-.089
2	3.702	10.888	31.091	-.204	.066	.134	-.148	-.342	-.039	.453
3	3.324	9.778	40.868	-.152	.420	.026	.478	.236	-.258	.008
4	2.968	8.729	49.597	-.068	-.012	.144	.161	-.499	-.474	-.255
5	2.257	6.638	56.235	-.276	-.331	-.077	-.578	.153	-.009	.017
6	2.099	6.174	62.409	.733	.018	.323	.247	.208	-.328	.479
7	2.037	5.991	68.401	.071	-.134	-.156	.057	.214	.349	.069
8	1.693	4.980	73.381	.035	.310	.433	-.224	-.331	.014	.108
9	1.424	4.189	77.569	.009	-.092	.064	.065	.113	.301	-.141
10	1.321	3.885	81.454	-.003	-.278	.033	.212	.206	.266	.024
11	1.111	3.269	84.723	.179	-.382	.083	.083	.256	-.116	.113
12	.984	2.895	87.617	-.110	-.162	-.174	-.178	-.053	-.068	.386
13	.882	2.595	90.213	.070	-.141	.202	.057	.230	.174	-.049

F8	F9	F10	F11	F12	F13	F14	F15	F16	F17	F18	F19	F20	F21	F22
.467	-.829	-.160	.430	-.141	-.125	-.145	.335	.140	-.660	-.159	-.464	.600	-.030	.133
.226	-.320	.344	-.122	-.190	-.315	-.393	.095	-.349	-.422	-.318	-.365	.539	.022	.481
-.098	.253	.140	.524	-.561	-.481	-.140	-.339	.255	-.052	-.008	.028	-.421	-.415	.306
-.017	.265	-.371	.528	.445	.214	.210	-.046	.149	-.200	.258	.386	.066	.560	.226
-.042	.641	.367	.463	.369	-.364	.350	.189	-.211	.178	-.325	.061	.095	.011	-.157
-.259	.224	.171	.094	.223	.290	-.078	.389	.131	-.121	-.209	-.341	-.404	.305	-.123
.153	.330	.323	-.100	-.020	.221	-.338	-.250	-.173	.495	.345	-.022	.101	.229	-.297
.242	.023	.213	-.131	.067	-.033	.555	-.306	.275	.403	-.073	-.164	-.091	-.051	.210
-.021	-.064	-.169	-.284	-.026	-.446	.076	.093	-.117	-.066	-.126	-.218	-.357	.313	.356
.066	.209	-.002	-.127	-.171	.107	.084	-.458	.153	-.237	-.260	.009	.265	.204	-.030
.126	-.014	-.283	.016	-.287	-.085	.185	.135	.070	.270	-.147	.297	-.115	-.332	-.102
.218	-.069	.254	.013	-.170	-.008	.069	.294	.060	-.160	.054	.127	-.083	.070	.041
-.053	.008	.157	.044	.004	.132	.188	.182	.048	-.234	.302	.039	.194	-.029	.356

F23	F24	F25	F26	F27	F28	F29	F30	F31	F32	F33	F34
-.371	-.446	-.395	-.605	-.696	-.282	-.349	.710	.688	.830	.551	.043
.487	.226	-.010	-.061	.646	.530	.228	.170	-.027	.132	.713	-.114
-.523	.222	-.318	.097	.466	-.295	-.128	-.030	.054	-.244	.396	.655
-.031	.162	-.284	.277	.167	.478	.307	.351	.259	.256	-.044	.196
.122	.033	.245	-.012	-.079	-.078	-.175	.208	.049	-.242	.264	.212
.104	.081	.243	-.063	.139	.069	-.122	-.234	.007	.042	.112	.077
-.214	-.393	-.147	.147	.344	.397	.001	-.032	-.033	.348	.147	.284
-.123	.018	.069	-.044	.062	-.049	-.103	-.042	-.205	.394	-.023	-.073
.154	-.179	.028	.372	-.090	.114	-.243	.163	.189	.127	-.183	.302
-.097	.557	.365	-.058	-.062	-.006	.170	.021	.039	.091	-.144	.141
.124	.038	.005	.379	.163	.081	.171	.215	-.037	.214	.170	-.325
-.105	.086	-.215	.146	-.102	-.063	.200	-.189	-.146	.134	-.327	.239
-.224	-.152	.040	.002	.073	.002	-.099	.199	-.263	-.147	.021	-.112

components from the above table for the further analysis. Likewise we can choose or take components according to our convenience. About 20% of the variation can be explained by the first component, 10% of the variation can be explained by the second component, and so on.

6. Conclusion

The main objective of this study is to find the significant factors of software development process which is affecting the reliability of the software. In this study two factors are added with the existing factors. 34 factors are considered for the analysis by relative weight method the significant factors are listed. Correlated factor are studied and listed. Principal component analysis is made for all the 34 factors. It may be useful for software developers and all those who related with software. If they implement this result they may get good and reliable software.

References

1. Barghout, M. and Littlewood, B.: A non-parametric order statistics software reliability model, *Journal of Software Testing, Verification and Reliability* **8(3)** (1998) 113-132.
2. Goel, A. L.: A software error detection model with applications, *in Proc. ACM Annu. Tech. Conf., ACM, Washington, DC* (1978) 496–500
3. Goel, A. L.: Software Reliability Models: Assumptions, Limitations and Applicability, *IEEE Transactions of Software Engg.,* **SE-11(12)** (1985) 1411–1423.
4. Goel, A. L. and Okumoto, K.: An analysis of recu.rrent software failures in a real-time control system,, *in Proc. ACM Annu. Tech. Conf., ACM, Washington, DC,* (1978) 496–500.
5. Jelinski, Z. and Moranda, P.: In Statistical Computer Performance Evaluation, in: *W. Freiberger, Ed. New York: Academic press* (1972) 465–484.
6. Littlewood, B. and Verrall, J. L.: A Bayesian reliability growth model for computer software, *Appl. Statist.,* **22** (1973) 332–346.
7. Patwa, S. and Malviya, K.: A Survey on Factors Affecting Testing Techniques in Object Oriented Software, *International Journal of Applied research on Information Technology and Computing,* **5** 2014 78–85.
8. Pham H.: System Software Reliability, *Springer Series in Reliability Engineering.*
9. Teng, X. and Pham, H.: A new methodology for predicting software reliability in the random field environments, *IEEE Transactions on Reliability* **55(3)** (2006) 458–468.
10. Zhang, X. and Pham, H.: An analysis of factors affecting software reliability, *Journal of Systems and Software* **50(1)**, (2000) 43–56.
11. Zhang, X. Shin, M. Y. and Pham, H.: Exploratory analysis of environmental factors for enhancing the software reliability assessment, *Journal of Systems and Software* **57** (2001) 73–78.
12. Zhang, X., Zhu M Y and Pham, H: A comparison analysis of environmental factors affecting software reliability, *Journal of Systems and Software* **109** (2015) 150–160.

A. Loganathan: Department of Statistics, Manonmaniam Sundaranar University, Tirunelveli, Tamilnadu - 627012, India
E-mail address: aln_24@rediffmail.com

R. Jeromia Muthuraj: Department of Statistics, Manonmaniam Sundaranar University, Tirunelveli, Tamilnadu - 627012, India
E-mail address: jeromstat@gmail.com