Experimental Analysis of Dependency Factors of Software Product Reliability using SonarQube

Sanjay L. Joshi, Bharat Deshpande, and Sasikumar Punnekkat

[1] Persistent Systems Limited, Goa, India Tel.: +91-20-66965051
sanjay_joshi@persistent.com
[2] BITS Pilani K K Birla Goa Campus, India Tel.: +91-832-2580438
bmd@goa.bits-pilani.ac.in
[3] School of Innovation, Design & Engineering, Mälardalen University, Sweden
Tel.: +46-21-107324 sasikumar.punnekkat@mdh.se

Abstract. Reliability is one of the key attributes of software product quality. Capability for accurate prediction of reliability will allow software product industry to have better market acceptability and enable wider usage in high integrity or critical applications domains for their product. Software Reliability analysis is performed at various stages during software product development life cycle. Popular software reliability prediction models proposed in literature are targeted to specific phases of life cycle with certain identified parameters. However, these models seem to have certain limitations in predicting software reliability in an accurate and acceptable manner to the industry.
A recent industrial survey performed by the authors identified several factors which practitioners perceived to have influence in predicting reliability. Subsequently we conducted a set of experiments involving diverse domains and technologies to validate the perceived influence of the identified parameters on software product reliability which was evaluated using SonarQube.
In this paper, we present our evaluation approach, experimental set up and results from the study. Through these controlled experiments and analysis of data, we have identified a set of influential factors affecting software reliability. This paper sets direction to our future research on modeling software product reliability as a function of the identified influential factors.

Keywords: Software Reliability · SonarQube · Empirical study · Experimental evaluation · Correlation · Software Product Attributes · Reliability prediction

1 Introduction

Quality is defined as “capability of a software product to conform to requirements.” as per ISO/IEC 9001[14]. According to ISO standard 25010 [15], the quality of product is defined as: “The totality of features and characteristics of a software product that bear on its ability to satisfy stated or implied needs”.

The focus in this paper is on one of the important attributes of quality, viz., the reliability of the software product [13].

In the past many models for predicting software reliability have been developed and studied extensively. However, these models are applicable under certain assumptions and for specific phases of development life cycle[9] [11]. Due to these limitations the proposed models have fallen short of gaining confidence with industry practitioners. An industrial survey was conducted by the authors to identify parameters that contributes to software reliability as perceived by industry professionals[8]. The survey highlighted that factors such as skill of developers and testers, design complexity, Commercial Off The Shelf (COTS) complexity, review efficiency contribute to reliability [8].

This paper evaluates the significance of such identified influential environmental [16]factors on software reliability for different software products in diverse domains and developed using diverse technologies. One of the popular open source tools, SonarQube was used in this study for evaluating software reliability for comparison purpose.

These experiments were performed in a large software development organization in India, which has laboratory setup necessary for performing such experiments. These laboratories basically serve as training centers for employees, both for new recruits as well as part of continuous learning. We have identified a set of independent input parameters [8] Refer Table 1 and dependent input parameters Table 2 for performing these experiments. Experiments were performed in a systematic and in controlled manner[1][2][4].

Paper organization: Section 2 discusses the background and some details regarding experimental context. Section 3 gives methodology followed for performing experiments. In section 4 the experimental results are presented along with discussion. Section 5 gives conclusion based on analysis done in previous section.

2 Background and Experimental Framework

In this study, we hypothesize reliability to be a function of defect leakage, post-delivery defects, schedule variance, effort variance, productivity, technology, commercial off the shelf(COTS) complexity, design complexity, unit test defects, integration test defects, system test defects, execution time and skill level of developer/ tester. These factors were identified as the most influential ones as perceived by stakeholders such as product users, coder, tester, designer, product managers, affecting software product reliability during the industrial survey conducted by the authors[8].

Experiments involved studying impact of these factors on reliability. Reliability is computed by varying one factor while maintaining all other factors constant. For example, skill level can be varied (from high to low) while keeping all other factors constant.

For each application, we performed minimum 30 combinations. For example: If skill level was identified as variable factor then all other factors such as tech-

Table 1. Independent Parameters (All discrete type)

Parameter	# of Levels	Levels	Comment
Skill	11	0-10	Technology specific skill based on internal evaluations
Design Complexity	4	Low, medium, high, very high	Based on expert judgments and complexity measures
Technology	3	C#, .NET, Sharepoint, ASP	C#, .NET used in one application and considered as one level
COTS Complexity	16	0-5 Simple 6-10 Medium 11-15 Complex	Complexity of COTS is based upon a. # of internal interfaces b. Impact factor c. # of calls through main program

Table 2. Dependent Parameters (all continuous type)

Parameter	Evaluated based on	Comment
UT Defects	No. of Defects during unit testing	Defects captured by coder
IT Defects	No. of Defects during Integration testing	Defects captured by tester / QA person
ST Defects	No. of Defects during system level testing	Defects captured by tester / QA person
Review efficiency	No. of defects captured in subsequent phases due to previous phases	Here, code review efficiency and test case review efficiency are taken into consideration
Post-delivery defects	No of defects- from site	Reported by site engineer.
On time (application execution time)	Execution time in hours	This is also known as operational time
Load	Number of parallel users	
Process metrics		
Schedule Variation-SV	Schedule	Schedule captured based on calendar days
Effort Variation-EV	Efforts	Effort captured in person hours
Productivity- P	Size & effort	Actual size is captured

nology, hardware, firmware, tools were kept constant. For four applications, we performed overall 120 experiments in the laboratory in which 560 people of different roles and skill levels participated. Reports on reliability [3] were obtained

Table 3. Overview of Applications

Application	Design Complexity	Domain	Technology	Platform
Risk Management	High	Project Management	C#, Sharepoint	Web based
eFinance	Medium	Finance	ASP.NET, Javascript	Web based
Photo zoom	Low	Entertainment	Jquery, Javascript	Mobile Application
ECG Management	Very High	Medical Device	C#, Sharepoint	Cloud and Mobile based

using SonarQube tool. SonarQube is popular open source platform for continuous inspection of code quality. The reliability figures from SonarQube are used as baseline for comparison purpose only. Hypothesis testing was used to check the statistical significance of the results.

In this section we also present reference of activities performed before entering experimentation area. Activities were targeted at software product reliability literature review and also conducting survey with practitioners and experts identified in the industry across globe. In the literature survey [9], we found that different reliability models are published in the past keeping Software Development Life Cycle (SDLC) as reference.

The realized software projects have been developed and managed as per ISO 9001:2015 standard and CMMI measurement and analysis, project monitoring and control issues. In these experiments, we captured data related to four products, which have been used for commercial purpose across the globe. All considered products can be classified as application in different domains and are listed in Table 3. Industrial standards proposed by Halstead were considered for categorizing design complexity of the applications[12].

For collecting data, we took help of different tools such as Jira, Rational Team Concert (RTC) and Team Foundation Server (TFS). These tools were used for collecting defects in requirement and design phase. Efforts and Schedule related data was captured using Microsoft Project Plan (MPP). We used GIT for configuration management and Rational Functional Tester (RFT) and Quick Test Professional (QTP) for testing automation. Other code quality related parameters were captured using PurifyPlus.

3 Methodology

Experimentation is powerful tool in software engineering. The main objective of performing experiments is to find cause and effect relationship [6]. Experiments

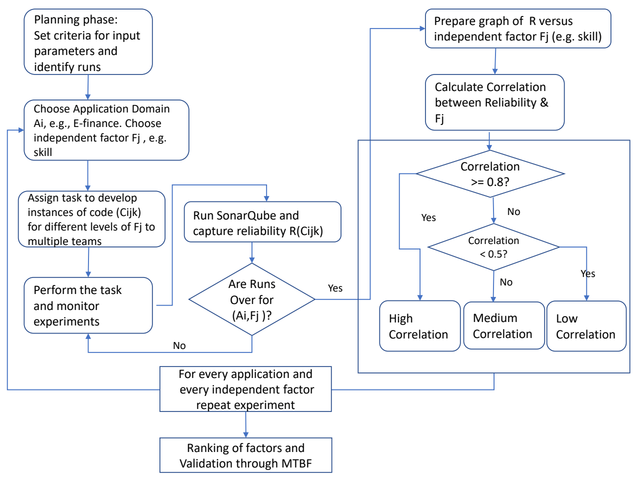

Fig. 1. Methodology Outline

were conducted in a multinational software product organization having centers across the globe. Series of experiments were conducted in controlled environment, where one parameter is considered as variable and other parameters are taken as constant [7].

The methodology used for performing the experiments is shown in Figure 1. For example, in experiments to study impact of skill on reliability, one functionality was identified of an application and task of developing it was assigned to software developers having varying skill levels. Minimum of 10K lines of code was the criteria set for developing the application. The design document was provided to all developers. Design complexity (input variable) along with other identified input parameters were kept constant. SonarQube was run on error free code to give the reliability factor for each skill level. To statistically conclude, more than 30 data points were recorded. By using this methodology, experiments were preformed for other identified factors.

4 Experimental Findings and Discussion

In this section we present our experiment findings and rank attributes influencing reliability. Chi-Square Test was used to statistically test whether parameters are having any impact on reliability [5]. We performed hypothesis test for each parameter separately using the R statistical tool.

Table 4 summarizes output of "R" for different skill levels for various technologies. In all above cases, probability value (p-value) for acceptance of null hypothesis is calculated and if the p-value is less than 0.05 then the null hypothesis is rejected. It can be seen from Table 4 that irrespective of the technology used, there is good correlation between skill level and reliability.

Table 4. Correlation between Reliability and Technology

Technology	Pearson's Chi-squared Test			Correlation
	χ^2	df	p-value	Is $p < 0.05$?
C#	393.75	336	0.0163	Yes
Sharepoint	441.15	344	0.000304	Yes
ASP.NET	196.00	144	0.002588	Yes
Java	226.65	180	0.0105	Yes

Fig. 2. Scatter Plot for Skill versus Reliability

As a sample, in Figure 2 we show the scatter plot of skill versus reliability. The pattern of the resulting points reveals that there exists correlation [10] between these two variables. To validate the data for other attributes, we performed χ^2 test and ANOVA for other skills, technologies and design / COTS complexity and confirmed their statistical significance.

Validation of results is also done using mean time between failure observed during testing and operational phases. This method is adopted for each application. Mean time between failure is judged based on operational discontinuity of an application. It is assumed that in case of critical, very high, high and medium type defects, application can behave differently and reliability of an application can be hampered. Sometimes during use, an application does not execute certain part of the code which can be a threat to the overall experimenting exercise. We covered maximum code during execution and checked all branches and nodes in the code to confirm the reliability figure. This is done through writing test cases for each branch and node identified for all features mentioned in the applications.

By careful design of the study and involving a broad spectrum and sufficiently large number of respondents across the organization, we have been able to eliminate most usual issues of external validity and reliability in empirical studies. However, one cannot rule out the possibility that we might have omitted yet another important factor from our list.

Table 5 summarizes the impact of other factors on reliability showing only absolute values of correlation factor. It can be concluded that reliability is strongly correlated with Skill factor, Post Delivery Defects and Review Efficiency, whilst reliability has good correlation with COTS complexity, System Test Defects,

Table 5. Correlation between reliability and different factors

Factor	C#	Share point	ASP. NET	Java/ Jquery	Inference	Average	Rank
Skill	0.890	0.9191	0.981	0.950	Strong	0.935025	2
UT Defects	0.231	0.040	0.224	0.233	No	0.182	9
IT Defects	0.295	0.230	0.262	0.260	No	0.26175	8
ST Defects	0.684	0.820	0.910	0.985	Good	0.84975	7
On time	0.040	0.201	0.040	0.105	No	0.0965	13
Load	0.833	0.789	0.980	0.820	Good	0.8555	6
Design Complexity	0.990	0.771	0.913	0.911	Good	0.89625	4
COTS Factor	0.846	0.843	0.921	0.900	Good	0.8775	5
Review Efficiency	0.997	0.910	0.870	0.960	Strong	0.93425	3
Post Deliv. Defects	0.936	0.990	0.960	0.966	Strong	0.963	1
SV	0.170	0.170	0.010	0.215	No	0.14125	11
EV	0.190	0.230	0.065	0.224	No	0.17745	10
Productivity	0.160	0.190	0.051	0.096	No	0.124275	12

Design Complexity and Load Condition. The last two columns show the average and rank of the factor based on its influence on reliability.

Current models are considering defects from field and internal in testing phase. However they are not considering factors like skill of developer and tester or review efficiency in development process.

5 Conclusions

One of the noteworthy findings from these experiments are factors like post-delivery defects, skill and review efficiency contributes significantly towards software product reliability and hence should be included in its prediction. With the help of this exercise, we could also eliminate (or at least keep on backstage) some parameters such as process metrics (Schedule Variance, Effort Variance and Productivity), Unit Test Defects, Integration Test Defect, System Test Defects. These experiments also indicate that load condition, Design complexity and COTS (in the order of increasing importance) could be significant in defining software product reliability. Though further detailing is needed, we consider this as a good starting point for defining an appropriate prediction model of software reliability.

6 Acknowledgement

We would like to thank Dr. Yogesh Badhe, data scientists from Persistent Systems Ltd. for his valuable support in data analysis. Also, we would like to thank Dr. Ramprasad Joshi for his valuable inputs for documentation while performing experiments. Punnekkat acknowledges support from FiC(SSF) Project.

References

1. Aleksandar Dimov, Senthil Kumar Chandran, Sasikumar Punnekkat, “How Do We Collect Data for Software Reliability Estimation?” , International Conference on Computer Systems and Technologies (CompSysTech), Sofia, pp: 155-160, 2010.
2. Walker, R. J, Briand, L. C, Notkin, D, Seaman, C. B, Tichy, W. F, Panel: empirical validation: what, why, when, and how. International Conference on Software Engineering(ICSE), Washington, DC, USA: IEEE Computer Society, 2003
3. Javier Garca-Munoz, Marisol Garca-Valls and Julio Escribano-Barreno, “Improved Metrics Handling in SonarQube for Software Quality Monitoring” , Distributed Computing and Artificial Intelligence, 13th International Conference pp 463-470, 2016.
4. Jedlitschka, A.; Pfahl, D.; Reporting Guidelines for Controlled Experiments in Software Engineering; ACM/IEEE Intern. Symposium on Software Engineering, Australia, 2005
5. Tore Dyb, Vigdis By Kampenes, Dag I.K. Sjberg, “A systematic review of statistical power in software engineering experiments” , Information and Software Technology, Volume 48, Issue 8, pp: 745-755, 2006
6. Kitchenham, B.A.; Pfleeger, S.L.; Pickard, L.M.; Jones, P.W.; Hoaglin, D.C.; El Emam, K.; Rosenberg, J.Preliminary guidelines for empirical research in software engineering; IEEE Transactions on Software Engineering, Vol. 28, No. 8 , Aug 2002, pp. 721 -734.
7. Kitchenham, B.A.; Hughes, R.T.; Linkman, S.G.; Modeling Software Measurement; IEEE Transactions on Software Engineering, Vol.27, No.9, September 2001
8. Sanjay L. Joshi, Bharat Deshpande, Sasikumar Punnekkat,“An Industrial Survey on Influence of Process and Product Attributes on Software Product Reliability” , NETACT, ISBN No:978-1-5090-6590-5, 2017.
9. Sanjay L. Joshi, Bharat Deshpande, Sasikumar Punnekkat, “Do Software Reliability Prediction Models Meet Industrial Perceptions?” , Proceedings of the 10th Innovations in Software Engineering Conference, Pages 66-73, 2017.
10. Maiwada Samuel and Lawrence Ethelbert Okey, “The Relevance and Significance of Correlation in Social Science Research” , International Journal of Sociology and Anthropology Research, Vol.1, No.3, pp.22-28, 2015.
11. Port, D.; Klappholz, D.: Empirical Research in the Software Engineering Classroom. Conference on Software Engineering Education and Training (CSEET), 2004
12. Kitchenham, B. A.; Pfleeger, S. L.; Pickard, L. M.; Jones, P. W.; Hoaglin, D. C.; Emam, K. E.; Rosenberg, J.: Preliminary guidelines for empirical research in software engineering. In: IEEE Trans. Softw. Eng. 28 (2002)
13. Claes Wohlin, Martin Höst, Per Runeson and Anders Wesslén, Software Reliability, Encyclopedia of Physical Sciences and Technology, Vol. 15, Academic Press, 2001
14. International Organisation for Standardization(ISO), ISO 9001: Quality Management Systems, 2015
15. ISO/IEC 25010:2011 : Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models
16. Zhu, M., Zhang, X., Pham, H. (2015). A comparison analysis of environmental factors affecting software reliability. Journal of Systems and Software, 109, 150-160