A METHODOLOGY FOR QUANTITATIVELY MANAGING THE CODING PHASE OF SOFTWARE DEVELOPMENT PROCESS

Boby John[1], Rajeshwar S Kadadevaramath[2]

[1]SQC & OR Unit, Indian Statistical Institute, Bangalore, India
[2]Department of Industrial Engineering and Management, Siddaganga Institute of Technology, Tumkur, India

Abstract

Many organizations use information technology to gain competitive advantage. As a result the demand for software products increased tremendously and the information technology industry has grown rapidly. As demand increased, the competition among information technology firms also increased. The information technology companies can no longer survive by just delivering the products but has to ensure the quality of products as well as the products have to be delivered on time without cost or effort overrun. Hence it is imperative for information technology companies to quantitatively manage the software development process. In fact quantitative project management is one of the requirements for achieving higher levels of capability maturity model. Lots of research has been carried out in the past to develop models to quantitatively manage the software development process. Most of these studies focussed on methodologies to quantitatively manage only one of the performance characteristics namely quality or schedule or effort. But to deliver the good quality software on time within the budgeted cost, all the critical performance parameters of software development process namely quality, productivity, effort, cost, etc need to be managed simultaneously. Many of these characteristics are related to each other and many cases the correlation is such that improving the performance of one characteristic will adversely affect the performance of other performance characteristics. Moreover all these performance characteristics need to be managed by controlling a common set of control parameters. Hence it is required to identify the best values of the control parameters which would simultaneously optimise all the performance characteristics. In this paper, the authors suggest a methodology to simultaneously optimize the performance characteristics of coding phase of the software development process. The same methodology can be used to simultaneously manage the different performance characteristics at other phases as well as the overall software development process. In this study the authors have taken two performance characteristics namely coding productivity and quality (measured in terms of defect density). The approach is to develop separate process performance models to estimate the coding productivity and defect density using the process parameters namely programmer skill, reviewer skill, review type, preparation time, module complexity and code review rate. Then the values of these process parameters which would simultaneously optimize the coding productivity and defect density are identified using Taguchi's loss function. The proposed approach has been implemented on seven software development projects and the results are very encouraging. Moreover the optimum obtained by the proposed method is much better than that of optimizing the coding productivity and defect density separately. The project managers also agreed that a common setting for process parameters which would optimize both performance characteristics simultaneously is much easier to implement than methods for managing different performance characteristics independently.

Keywords: *Quantitative project management, multi response optimization, dummy variable regression, Taguchi's loss function*

---***---

1. INTRODUCTION

The organizations need to have the competitive advantage to succeed in globalised business world. Brand image, pricing, technology, skilled resources, etc (Samson & Terziovski 1999; Adam et al 2001) can provide the aforementioned competitive advantage. Recently more and more organizations are depending on Information Technology (IT) to gain the competitive advantage. As the demand for the software products increased, the number of firms also increased. But even today many IT companies struggle to deliver quality software on time within the budgeted cost. So the software companies need to have a balanced approach of management for the different sub process of software development life cycle to meet the goals or requirements of quality, schedule, cost, etc. The Capability Maturity Model developed by Software Engineering Institute (SEI) of the Carnegie Mellon University classified the software processes into five maturity levels namely initial, repeatable, defined, managed and optimizing (Paulik et al 1993). The research has shown that higher maturity levels are associated with higher quality, better on time delivery and lower cost (Harter et al 2000). Still most of the studies on quantitatively managing the software processes are focusing on only one performance characteristic or response variable and majority of them are on only software quality.

Tamura (2009) has developed three process performance models, one for estimating product quality objectives (defect density) in terms of requirement inspection rate (pages per hour) and whether prototype is developed or not as factors. The second one is for estimating code review yield (% of defects present in the software that are removed by the review) in terms of review rate (the number of lines of code reviewed per hour). The third model is to predict the escaped unit test defect density with test coverage as controllable variable. Hao and Zhang (2011) also developed a model for delivered defect density using average team skill level and test coverage as controllable variables. Similarly lots of models have been developed for predicting software defects or classifying the software as defect prone or not. These models are generally developed using either statistical techniques (Nagappan et al 2005; Khoshgoftaar and Allen 1999; Cruz and Ochimizu 2009; Bibi et al 2006; Binkley 2007; Schneidewind 2001; Sandhu et al 2010) or machine learning techniques (Tao and Wei-hua 2010; Fenton et al 2007; Ceylan et al 2006; Kaur and Malhotra 2008; Elish and Elish 2008; Afzal and Torkar 2008; Zhu and Wu 2009; Hribar and Duka 2010).

Studying the relationship between software quality or defect density with process control variables and developing a prediction model is useful for achieving the quality goals. Using the model, the project managers can identify the optimum values of the control variables which would bring the quality close to the target. But achieving quality goals or target alone is not sufficient. The organizations also need to manage the software development life cycle process to achieve the goals of other performance characteristics namely productivity, schedule, cost etc. Many of these performance characteristics may be correlated. Moreover the managers need to adjust a common set of controllable variables to achieve the goals on multiple characteristics. Hence it is desirable to have a common setting of controllable variables which would simultaneously optimize the multiple performance characteristics. The same problem exists in other industries also and lot of work has been carried out in simultaneous optimization of multiple performance or output characteristics. Based on the success of simultaneous optimization of multiple responses in manufacturing industries, the authors suggest a similar methodology for achieving simultaneously the goals of multiple performance characterstics in IT industry. This paper demonstrates the proposed approach for simultaneously achieving the targets on quality and productivity at coding phase of software development life cycle. The coding phase includes development and code review sub processes. The quality and productivity are studied in this paper, the same approach can be used for other performance characteristics also. In fact, the approach can be used for simultaneously achieving the goals on any number of characteristics. The approach can be used for other sub processes like design, testing, etc and also for quantitatively managing the entire software development process.

The remaining part of the paper is organised as follows: a brief discussion on simultaneous optimisation of multiple characteristics is given in section 2, Taguchi’s loss function is explained in section 3, section 4 describes the proposed methodology for quantitatively managing the coding phase of software development life cycle and the results and conclusions are discussed in section 5.

2. SIMULTANEOUS OPTIMIZATION OF MULTIPLE PERFORMANCE CHARACTERISTICS

Many industrial processes are characterised by more than one performance characteristics. For example, in surface hardening process not only the surface hardness, the case depth can also be important. Similarly in machining process, not only the dimensional accuracy, the surface finish also need to be optimised. Lot of studies have been carried out in the past on developing methodologies for simultaneous optimisation of multiple performance characteristics. Montgomery and Castillo (1993) suggested response surface methodology. Harrington (1965) and Derringer (1994) developed the desirability function approach. Koksoy and Yalcinoz (2006) used mean square error criterion for analysing several quality characteristics simultaneously. Su and Tong (1997) suggested principal component analysis based methodology for multiple characteristics optimization. Hsu (2004) proposed an integrated approach based on neural networks, exponential desirability functions & tabu search. Many researchers have also used data envelopment analysis (Liao 2004), fuzzy logic (Antony et al 2006) and gray relational analysis (Saha and Mandal 2013) for simultaneous optimization of multiple performance characteristics. Logothetis and Haigh (1988), Tong et al (1997), Magsoodloo and Chang (2001), Reddy et al (1998), Boby (2012) and Wu (2002) used methods based on Taguchi's loss function (Taguchi et al 1989) for simultaneous optimization of multiple performance characteristics.

Lot of case studies are also available in the literature on multiple characteristic optimization. Fung and Kang (2005) optimized the injection-moulding process for friction properties of fiber-reinforced polybutylene terephthalate using Taguchi method and principal component analysis. Dubay and Yadava (2008) presented a hybrid of Taguchi and response surface method for the multi-response optimisation of a laser beam cutting process. Gauri and Chakraborty (2009) suggested a modified principal component analysis based method for multi-response optimisation of wire electrical discharge machining process. Asilturk and Neseli (2012) presented Taguchi method-based response surface methodology to determine multi-objective optimal cutting conditions for CNC turning process. Wei and Yuying (2008) applied Pareto-based multi-objective genetic algorithm to optimize sheet metal forming process.
In this research, the authors are also proposing Taguchi's loss function methodology for simultaneously achieving the quality and productivity goals at the coding phase of software development process.

3.TAGUCHI'S LOSS FUNCTION APPROACH

According to Taguchi, any deviation of the performance characteristic from the target needs to be treated as loss. He developed the quadratic loss function to quantify the loss.

Taguchi's quality loss function (Fowleks and Creveling 1998) is defined as

$$L(y) = k(y - T)^2 \qquad (1)$$

where y is the performance characteristic, T is the target value of the performance characteristic and k is a proportionality constant namely quality loss coefficient. The value of k can be chosen based on economical considerations. When the characteristic y is on the target T, there will be no loss and as the performance characteristic y deviates from the target, the loss increases. The proportionality constant k can also be chosen to ensure that as long as the performance characteristic is meeting the service level agreements (SLA) or specification limits, then the loss will be ≤ 1. This can be achieved by choosing k as

$$k = \left(\frac{2}{USL - LSL} \right)^2 \qquad (2)$$

where USL is upper specification limit & LSL is the lower specification limit of the performance characteristic. The aforementioned choice of k ensures that the loss will be equal to 1 when the response variable is on either of the specification limits. The loss will be < 1 when the performance characteristic is within the specification limits and the same will be 0 when the characteristic is on the target. Similarly k can be chosen for one sided SLA or specification limits also.

The quality loss function is defined for three types of response variables namely smaller the better (STB), larger the better (LTB) and nominal the best (NTB). Let $y_1, y_2, - - -, y_n$ be the n observations of the performance characteristics y. Then the formula for computing the different loss functions are as follows:

For nominal the best characteristics

$$l(y) = \frac{1}{n} k \sum_{i=1}^{n} (y_i - T)^2 \qquad (3)$$

For smaller the better characteristics

$$l(y) = \frac{1}{n} k \sum_{i=1}^{n} y_i^{\,2} \qquad (4)$$

And for larger the better characteristics

$$l(y) \frac{1}{n} k \sum_{i=1}^{n} \frac{1}{y_i^2} \qquad (5)$$

Finally the overall expected loss $L(y)$ is computed as the average of the expected losses of individual response variables as given in (6).

$$L(y) = \frac{1}{p} \sum_{i=1}^{p} l(y_i) \qquad (6)$$

The combination of the process control variables with minimum overall expected loss would simultaneously optimise all the performance characteristics

4. METHODOLOGY FOR QUANTITATIVELY MANAGING THE CODING PROCESS

The application of simultaneous optimization of multiple performance characteristics for software development process is described in this session. Taguchi's loss function methodology is used in this study. The study is carried out at the coding phase of the software development life cycle. The performance characteristics chosen for simultaneous optimisation are code review defect density and coding productivity. The defect density is a measure of quality and is defined as the number of defects per unit size (Fenton and Bieman 2014). The unit size generally is taken as 1000 lines of code. The coding productivity is measured as the ratio of size over effort. In other words, coding productivity is the number of lines coded per unit effort. The specifications on the performance characteristics under study are given in table 1.

Table 1: Performance characteristics with specification

Sl No	Performance Characteristics	Target	LSL	USL
1	Defect density	1.0	0.5	1.5
2	Coding productivity	12	10	14

The specification on coding productivity is arrived based on the industry benchmark for the underlying technology. Ideally a software should be free from any defect or bug. But since the software development is a human activity, errors can occur and bugs can get injected during designing, coding or integrating the modules, etc. Hence it is important to detect and remove as many bugs as possible before releasing the software. It is even better to detect the bugs at the early stages of software development like design review or code review than down the line at system or acceptance testing. So if the code review defect density target is zero, then the review may not be carried out properly and many defects may escape to subsequent phases. Similarly if the defect density target is very high and the programmers are highly skilled and knowledgeable, then it becomes almost impossible to achieve code review defect density targets. Hence based on the past performance the specification on defect density is arrived at as given in table 1.

The discussions with the project managers and engineers of the firm revealed that mostly the people related factors are easier to control in software development by changing the composition of the development as well as review teams. Moreover these factors are likely to impact the performance characteristics as software development is a human activity. The controllable factors identified for the study is given in table 2.

Table 2: List of controllable factors

SL No	Factor Name	Description
1	Programmer skill	0: Fresher, 1:Experienced
2	Reviewer skill	0: Fresher, 1: Experienced
3	Review type	0: Peer review, 1: Fagan review
4	Preparation time	0: 20% of total time, 1: 30% of total time
5	Complexity	0: Simple, 1: Complex
6	Code review rate	0: 75 to 100 lines per hour, 1: 50 to 75 lines per hour

The data on the factors and the performance characteristics namely defect density and coding productivity are collected from the past projects. Two models, one each for defect density and coding productivity are developed using the factors as independent variables. Since the factors are categorical, dummy variable regression is used for developing the models. The regression statistics, regression Anova table and coefficient table for defect density model is given in table 3 to 5 respectively.

Table 3: Defect density regression model statistics

Statistics	Value
R^2	0.955
Adjusted R^2	0.925
Standard Error	0.5524

Table 4: Defect density regression Anova table

	df	SS	MS	F	p value
Regression	6	59.0193	9.8365	32.2352	0.0000
Residual	9	2.7463	0.3051		
Total	15	61.7656			

Table 5: Defect density model coefficient table

	Coefficients	Standard Error	t Stat	P-value
Intercept	2.33078	0.36538	6.37904	0.00013
Programmer Skill	-1.81519	0.27620	-6.57199	0.00010
Reviewer skill	1.02766	0.27620	3.72070	0.00477
Review type	-1.74885	0.27620	-6.33178	0.00014
Preparation time	2.44989	0.27620	8.86993	0.00001
Complexity analysis	1.14978	0.27620	4.16284	0.00244
Code review rate	0.14612	0.27620	0.52905	0.60957

The table 3 shows that R^2 and adjusted R^2 are reasonably high (> 0.6) and the p value in the regression anova table (refer table 4) < 0.05, hence the model is significant. The table 5 shows that except code review rate, other factors are significant at 5% level (p value < 0.05). Hence the regression analysis is carried out again by dropping the code review rate factor. The new coefficient table is given in table 6.

Table 6: Modified coefficient table for defect density model

	Coefficients	Standard Error	t Stat	P-value
Intercept	2.40384	0.32587	7.37669	0.00002
Programmer Skill	-1.81519	0.26607	-6.82222	0.00005
Reviewer skill	1.02766	0.26607	3.86236	0.00315
Review type	-1.74885	0.26607	-6.57286	0.00006
Preparation time	2.44989	0.26607	9.20765	0.00000
Complexity	1.14978	0.26607	4.32134	0.0015099

Form table 6, the model for estimating the defect density is

$$\textit{Defect density} = 2.40 - 1.82\ \textit{Programmer skill} + 1.03\ \textit{Reviewer skill} - 1.75\ \textit{review type} + 2.45\ \textit{Preparation time} + 1.15\ \textit{Complexity} \quad (7)$$

Similarly the regression statistics, regression Anova table and coefficient table for coding productivity model is given in table 7 to 9 respectively

Table 7: Productivity regression model statistics

R^2	0.913
Adjusted R^2	0.853
Standard Error	1.376

Table 8: Productivity regression Anova table

	df	SS	MS	F	p value
Regression	6	178.63	29.77	15.72	0.00
Residual	9	17.04	1.89		
Total	15	195.67			

Table 9: Productivity model coefficient table

	Coefficients	Standard Error	t Stat	P-value
Intercept	13.53	0.9102	14.8649	0.0000
Programmer Skill	3.57	0.6880	5.1886	0.0006
Reviewer skill	-0.2225	0.6880	-0.3234	0.7538
Review type	0.37	0.6880	0.5378	0.6038
Preparation time	-1.405	0.6880	-2.0420	0.0715
Complexity	-5.195	0.6880	-7.5504	0.0000
Code review rate	-1.6625	0.6880	-2.4163	0.0388

The table 7 shows that R^2 and adjusted R^2 are reasonably high (> 0.6) and the p value in the regression Anova table (refer table 8) < 0.05, hence the model is significant. The table 9 shows that except review type and reviewer skill, other factors are significant at 5% level (p value < 0.05). Hence the regression analysis is carried out again by dropping the reviewer skill and review type factors. The new coefficient table is given in table10.

Table 10: Modified coefficient table for productivity model

	Coefficients	Standard Error	t Stat	P-value
Intercept	13.604	0.711	19.137	0.000
Programmer Skill	3.570	0.636	5.615	0.000
Preparation time	-1.405	0.636	-2.210	0.049
Complexity	-5.195	0.636	-8.170	0.000
Code review rate	-1.663	0.636	-2.615	0.024

Hence the model for coding productivity is

Coding productivity = 13.604 +3.57 Programmer skill – 1.405 Preparation time -5.195 Complexity - 1.663 Code review rate

(8)

The defect density and coding productivity are computed for all the possible combination of factor values using (7) and (8). Since there are 6 factors and each can take 2 values, there are 64 possible combinations. Then the expected loss for defect density, productivity and the overall expected loss for all these combinations are computed using (3) and (6). The combination of factor values which would minimise expected loss is given in table 11.

Table 11: Optimum combination

Factor	Value
Programmer Skill	Experienced
Reviewer skill	Experienced
Review type	Fagan review
Preparation time	20% of total time
Complexity	Complex
Code review rate	75 to 100 lines per hour
Defect density	1.017
Productivity	12.574
DD_loss	0.001
Productivity_Loss	0.082
Expected Overall loss	0.042

The table 11 above shows that the estimated defect density for optimum setting or combination of factors would be 1.017, very close to the target of 1 and well within the specification limits. Similarly the estimated productivity for optimum setting or combination of factors would be 12.574, very close to the target of 12 and well within the specification limits (refer table 1). The optimum combinations of factors obtained by optimising performance characteristic defect density also gave the same combination but that of optimising productivity alone has given different combinations. In fact there are four different combinations that would bring productivity very close to the target. The combinations are given in table 12.

Table 12: Factor combinations for optimising productivity alone

Combinations	1	2	3	4
Programmer Skill	Experienced	Experienced	Experienced	Experienced
Reviewer skill	Fresher	Fresher	Experienced	Experienced
Review type	Peer review	Fagan review	Peer review	Fagan review
Preparation time	20%	20%	20%	20%
Complexity	Complex	Complex	Complex	Complex
Code review rate	50 - 75 lines	50 - 75 lines	50 - 75 lines	50 - 75 lines
Defect Density	4.19	2.44	5.22	3.47
Productivity	11.91	11.91	11.91	11.91

The table 12 revealed that even though the expected productivity is 11.91, which is very close to the target of 12 but in all the four cases expected defect density is not meeting the SLA or specification. Hence it is better to simultaneously optimise the multiple performance characteristics. Moreover the project managers can achieve the targets on different performance characteristic with a common setting of process control variables or factors. The managers can also easily identify the second best, third best combinations, etc. This would give them lot of options to quantitatively manage the process.

The approach is validated by comparing the results of seven modules satisfying the optimum combination of factors. The defect density and coding productivity of the seven modules used for validation is given in table 13.

Table 13: Validation results

Module id	Defect density	Coding Productivity
1	1.05	12.85
2	0.98	13.21
3	1.24	12.01
4	1.18	11.93
5	0.97	11.86
6	1.4	12.67
7	1.32	13.01

The table 13 shows that for all the seven modules, the defect density is close to the predicted defect density of 1.017 for the optimum combinations. Similarly all the seven productivity values are also close to the predicted value of 12. 574. Hence it is decided to recommend the proposed methodology for quantitatively managing the coding phase of all the upcoming projects.

5. CONCLUSION

The software industry has been witnessing tremendous growth. Still delivering the quality software without cost or schedule overrun is a challenge for many information technology companies. It is required to quantitatively manage the software development process to achieve the goals on software quality, productivity, cost, etc. But most of the published works are on quantitatively managing the process to achieve goals of only one performance characteristics. In this paper, the authors suggested a methodology to simultaneously achieve the goals of multiple performance characteristics. The paper discussed a special case of simultaneously optimizing the defect density and coding productivity of the coding phase of the software development process.

Through discussions with the technical experts, the different factors influencing the defect density and productivity are identified. Then two models has been developed one each for estimating the defect density and productivity. The models are developed using dummy variable regression. Then using Taguchi's loss function approach the optimum combination of factors that would simultaneously bring both defect density and productivity close to the respective targets are identified. The methodology would help the project managers to meet the requirements of multiple performance characteristics with common settings of factors. It is also found that the optimum combination obtained through the suggested methodology is superior to that of optimising individual performance characteristics separately. The suggested methodology can also be used for simultaneously achieving goals of multiple performance characteristics of other sub processes like design, testing, etc of the software development life cycle as well as the entire software development process itself.

REFERENCES

[1] Adam EE, Flores BE and Macias A (2001) Quality improvement practices and the effect on manufacturing firm performance: evidence from Mexico and the USA. International Journal of Production Research 39(1): 46 – 63.

[2] Afzal W and Torkar R (2008) A comparative evaluation of using genetic programming for predicting fault count data. In Third IEEE International Conference on Software Engineering Advances: 407-414.

[3] Antony J, Anand RB, Kumar M and Tiwari MK (2006) Multiple response optimization using Taguchi methodology and neuro-fuzzy based model. Journal of Manufacturing Technology Management 17(7): 908–925.

[4] Asiltürk I and Neşeli S (2012) Multi response optimisation of CNC turning parameters via Taguchi method-based response surface analysis. Measurement 45(4):785-794.

[5] Bibi S, Tsoumakas G, Stamelos I and Vlahavas IP (2006) Software Defect Prediction Using Regression via Classification. In AICCSA: 330-336.

[6] Binkley D, Feild H, Lawrie D and Pighin M (2007) Software fault prediction using language processing. Testing: IEEE Academic and Industrial Conference Practice and Research Techniques: 99-110.

[7] Boby John (2012) Simultaneous optimization of multiple performance characteristics of carbonitrided pellets: a case study. The International Journal of Advanced Manufacturing Technology 61(5-8): 585-594

[8] Ceylan E, Kutlubay FO and Bener AB (2006) Software defect identification using machine learning techniques. In 32nd IEEE EUROMICRO Conference on Software Engineering and Advanced Applications: 240-247

[9] Cruz AC and Ochimizu K (2009) Towards logistic regression models for predicting fault-prone code across software projects. In 3rd IEEE International Symposium on Empirical Software Engineering and Measurement: 460-463.

[10] Del Castillo E and Montgomery DC (1993) A nonlinear programming solution to the dual response problem. Journal of Quality Technology 25(3): 199 - 204.

[11] Derringer G (1994) A balancing act: Optimising product's properties Quality Progress 27(6): 51- 58.

[12] Dubey, AK and Yadava V (2008) Multi-objective optimisation of laser beam cutting process. Optics & Laser Technology 40(3):562-570.

[13] Elish KO and Elish MO (2008) Predicting defect-prone software modules using support vector machines. Journal of Systems and Software 81(5): 649-660.

[14] Fenton N and Bieman J (2014) Software metrics: a rigorous and practical approach, CRC Press.

[15] Fenton N, Neil M, Marsh W, Hearty P, Marquez D, Krause P and Mishra R (2007) Predicting software defects in varying development lifecycles using Bayesian nets. Information and Software Technology 49(1): 32-43.

[16] Fowleks WY, Creveling CM (1998) Engineering methods for robust product design: using Taguchi methods in technology and product development. Addison-Wesley Longman Inc. USA

[17] Fung CP and Kang PC (2005) Multi-response optimization in friction properties of PBT composites using Taguchi method and principle component analysis. Journal of materials processing technology 170(3):602-10.

[18] Gauri SK and Chakraborty S (2009) Multi-response optimisation of WEDM process using principal component analysis. The International Journal of Advanced Manufacturing Technology 41(7-8):741-748.

[19] Hao Y and Zhang YF (2011) Statistical prediction modeling for software development process performance. In 3rd IEEE International Conference on Communication Software and Networks (ICCSN): 703-706.

[20] Harrington E (1965) The desirability function Industrial Quality Control 21(10): 494 – 498.

[21] Harter DE, Krishnan MS and Slaughter SA (2000) Effects of process maturity on quality, cycle time, and effort in software product development. Management Science 46(4): 451- 466.

[22] Hribar L and Duka D (2010) Software component quality prediction using KNN and Fuzzy logic. In Proceedings of the 33rd IEEE International Convention MIPRO: 402-408.

[23] Hsu CM (2004) An integrated approach to enhance the optical performance of couplers based on neural networks, desirability functions and tabu search. International Journal of Production Economics 92(3):241 – 251.

[24] Hung-Chang Liao (2004) A data envelopment analysis method for optimising multi-response problems with censored data in the Taguchi method. Computers and Industrial Engineering 46(4): 817-835.

[25] John Boby (2013) Application of desirability function for optimizing the performance characteristics of carbonitrided bushes. International Journal of Industrial Engineering Computations 4(3): 305-314.

[26] Kaur A and Malhotra R (2008) Application of random forest in predicting fault-prone classes. In IEEE International Conference on Advanced Computer Theory and Engineering: 37-43.

[27] Khoshgoftaar TM and Allen EB (1999) Logistic regression modeling of software quality. International Journal of Reliability, Quality and Safety Engineering 6(4): 303-317.

[28] Koksoy O and Yalcinoz T(2006) Mean square error criteria for multi-response process optimisation by a new genetic algorithm Applied Mathematics and Computations 175(2): 1657 – 1674.

[29] Logothetis N and Haigh A (1988) Characterizing and optimizing multi-response processes by Taguchi method. Quality and Reliability Engineering International 4(2):159–169.

[30] Maghsoodloo S and Chang CL (2001) Quadratic loss functions and signal to noise ratios for bivariate response. Journal of Manuf acturing Systems 20(1):1– 12

[31] Nagappan N, Williams L, Osborne J, Vouk M and Abrahamsson P (2005) Providing test quality feedback using static source code and automatic test suite metrics. In 16th IEEE International Symposium on Software Reliability Engineering: 10-pp.

[32] Paulk MC (1993) Comparing ISO 9001 and the capability maturity model for software. Software Quality Journal 2(4): 245-256.

[33] Reddy PBS, Nishina K and Babu AS (1998) Taguchi's methodology for multi-response optimization—a case study in the Indian plastics industry. International Journal of Quality & Reliability Management 15(6):646–668.

[34] Samson D and Terziovski M (1999) The relationship between Total Quality Management practices and operational performance. Journal of Operations Management 17(4): 393 – 409.

[35] Sandhu PS, Kaur M and Kaur A (2010) A Density Based Clustering approach for early detection of fault prone modules. In 2010 IEEE International Conference on Electronics and Information Engineering 2: V2-525.

[36] Schneidewind NF (2001) Investigation of logistic regression as a discriminant of software quality. In Proceedings of IEEE Seventh International Software Metrics Symposium: 328-337.

[37] Su CT and Tong LI (1997) Multi-response robust design by principal component analysis. Total Quality Management 8(6): 409 – 416.

[38] Taguchi G, Elsayed EA and Hsiang T (1989) Quality engineering in production systems. McGraw-Hill, New York

[39] Tamura S (2009) Integrating CMMI and TSP/PSP: Using TSP Data to Create Process Performance Models (No. CMU/SEI-2009-TN-033), Carnegie-Mellon University, Software Engineering Institute, Pittsburgh.

[40] Tao W and Wei-hua L (2010) Naive bayes software defect prediction model. In IEEE International Conference on Computational Intelligence and Software Engineering: 1-4.

[41] Tong LI, Su CTand Wang CH (1997) The optimization of multiresponse problems in the Taguchi method. International Journal of Quality & Reliability Management 14(4):367–380

[42] Wei L and Yuying Y (2008) Multi-objective optimization of sheet metal forming process using Pareto-based genetic algorithm. Journal of materials processing technology 208(1):499-506.

[43] Wu FC (2002) Optimization of multiple quality characteristics based on percentage reduction of Taguchi's quality loss. International Journal of Advanced Manufacturing Technology 20(1):749–753

[44] Zhu D and Wu Z (2009) The Application of Gray-Prediction Theory in the Software Defects Management. In IEEE International Conference on Computational Intelligence and Software Engineering: 1-5.