Contents lists available at GrowingScience

Journal of Project Management

homepage: www.GrowingScience.com

Application of multistage process control methodology for software quality management

Boby John[a*], R. S. Kadadevaramath[b] and I. A. Edinbarough[c]

[a]*SQC & OR Unit, Indian Statistical Institute, 8th Mile, Mysore Road, Bangalore, Karnataka State, India – 560 059*
[b]*Department of Industrial Engineering & Management, Siddaganga Institute of Technology, Tumkur, Karnataka State, India – 572103*
[c]*Department of Manufacturing and Industrial Engineering, University of Texas at Brownsville, USA*

C H R O N I C L E

Article history:
Received: October 1, 2016
Received in revised format: November 16, 2016
Accepted: February 24, 2017
Available online:
February 24, 2017

Keywords:
Quantitative project management
Defect density
Classification and regression tree
Ridge regression
Multistage process control

A B S T R A C T

As the need for software increased, the number of software firms and the competition among them also increased. The software companies in developing countries like India can no longer survive based on cost advantage alone. The firms need to deliver competitively priced quality software products on time. This can be achieved through quantitatively managing the different phases or sub processes in software development process. But quantitative management of a process consisting of a set of interlinked sub processes or stages with the output of one sub process influencing that of subsequent stages and final output is not easy. The process performance models developed for quantitative management of software development process often model the final outcome in terms of factors from various stages together or focuses only on quantitatively managing a particular sub process independently. In manufacturing and other engineering industries, the processes with multiple sub process are monitored and controlled using multistage process control methodology. This paper is an application of multistage statistical process control for managing the software development process. The suggested methodology is a combination of process performance models and control charts. The proposed methodology can be easily implemented for controlling various types of software projects like development projects, incremental development projects, testing projects etc. The methodology also provides the project manager the opportunity to tighten or relax the control at various sub processes based on the project team's strengths and still achieve the goal on the final outcome.

1. Introduction

Many organizations utilize information technology (IT) or use automation to gain the business advantage over the competitors (Adam et al., 2001; Samson & Terziovski, 1999; Asher & Kanji, 1999). As a result, the IT industry has grown rapidly in the recent past. As the number of software firms increased, the competition among the companies also increased. The software companies in countries like India can no longer survive or satisfy the customer by cost advantage alone. The organizations need to deliver quality software products on time at a competitive cost. The studies have shown that the software quality, development cycle time and effort are related (Harter et al., 2000). The software

* Corresponding author. Tel.: +91 94487 04182
E-mail address: boby@isibang.ac.in; bobymon@outlook.com (B. John)

2017 Growing Science Ltd.
doi: 10.5267/j.jpm.2017.2.001

quality is also related to customer satisfaction (Prazinger & Nath, 2000). The studies have also shown that higher the CMM level, better is the software quality (Herbsleb et al., 1997).

Defining software quality is not easy and there is no single adequate measure for software quality. ISO 9126 standard (1991) defined software quality as the totality of features and characteristics of a software product that bear on its ability to satisfy the stated and implied needs of the customer. The Capability Maturity Model (CMM) of Software Engineering Institute (SEI) of Carnegie Mellon University classifies the software process into five maturity levels namely initial, repeatable, defined, managed and optimized (Paulik et al., 1994; Pressman, 2005). For quantifying and monitoring software development process, software quality is often measured in terms of delivered defect density (Fenton & Bieman, 2014). The delivered defect density is the number of defects per unit size. The widely accepted approach for software quality management is to set the target or goal on delivered defect density and then manage the various phases or sub process in the software development life cycle to achieve the set goal. The project managers generally utilize their industry experience, software engineering knowledge, process performance models (Tamura, 2009; Hao & Zhang, 2011, John & Kadadevaramath, 2015), defect prediction models, etc. to implement the aforementioned approach. A wide variety of defect prediction models is available in the literature. These models are developed for either predicting the defects or classifying the software module as defect prone or not. Most of them are based on either statistical learning techniques (Niel, 1992; Turhan & Bener, 2007) or machine learning techniques (Ceylan et al., 2006; Song et al., 2006). The defect prediction models often use static code attributes like code complexity, etc. as predictors and use difficult to change factors. Hence these models are more suitable for prediction than process control and monitoring. The software quality also depends on people-related factors like programmer skill, domain knowledge, experience, etc. (Antony & Fergusson, 2004). Moreover, the defect density at different sub processes or phases in software development lifecycle may be related to that of subsequent phases and also to the delivered defect density. So one of the ways to achieve the goal on delivered defect density can be to link the delivered defect density with phase wise defect densities and then control the different phases in the development process to meet the intermediate goals set on phase wise defect densities. This can be done using multistage statistical process control. The engineering and chemical industries have been successfully using multistage statistical process control methodology for monitoring multistage processes. The software development process also can be considered as a multistage process and the different phases of design, coding, testing, etc. can be considered as multiple stages in the process. This paper is an application of multistage statistical process control methodology for monitoring and controlling software development process.

The remaining part of this paper is organized as follows: a brief description of multistage statistical process control is given in session 2, the proposed methodology for controlling the software development life cycle process is given in session 3, session 4 discusses the application of proposed methodology in software quality management and the conclusions are given in session 5.

2. Multistage statistical process control

Many manufacturing and service delivery processes consist of multiple stages or sub processes. The examples are semiconductor manufacturing, software development, automotive body assembly, etc. (Tsung et al., 2008). Such processes are called multistage processes. The multistage process output quality often depends on the quality at the intermediate stages. Hence to achieve the output quality goal or target, it is necessary to link the final quality with the quality at the intermediate stages and control the intermediate stages in the process. The multistage process control techniques are developed for the aforementioned purpose. There are two widely used approaches for multistage process control namely engineering process control and statistical process control. The engineering process control uses a linear state space model based on engineering knowledge and physical laws (Jin & Shi, 1999; Ding et al., 2002; Djurdjanovic & Ni, 2001). The state space model provides an engineering tool for analyzing, modeling and controlling multistage process.

The commonly used multistage statistical process control methods are regression adjustment approach (Hawkins, 1993; Shu et al., 2004) and cause selecting chart (Shu et al., 2003; Shu & Tsang, 2003; Shu et al., 2005). The logic of regression adjusted approach and cause selecting chart are very similar to that of model-based control charts. The model-based approach is commonly used for autocorrelated data. The approach is to fit a suitable time series model to the quality characteristic and then monitor the residuals using a control chart (Montgomery, 2007). In regression adjustment approach, regression models are fitted for stage wise quality characteristic with control variables from respective stages as predictor variables. Then the residuals of each model are plotted on univariate control charts. In cause selecting chart, the regression models for the quality characteristic at every stage is developed by taking the quality characteristic at the previous stage as the only predictor variable. Then the residuals of the model are plotted on a suitable control chart. Another suggestion for monitoring multistage processes is to fit regression models for quality characteristics at different stages and then monitoring the residuals of the models using CUSUM charts (Zantek et al., 2006).

In this paper, the authors describe the application of regression adjustment approach for monitoring quality of software development process.

3. Methodology

The step by step details of the proposed multistage statistical process control methodology for managing quality during software development process is given below. The quality is measured in terms of defect density.

Step1: Classify the software development projects into homogenous groups. The groups should be formed in such way that the projects within a group are similar to each other but are dissimilar to projects in other groups. The domain, technology, account, etc. can be the grouping variable.

Step 2: For every group, identify the different sub processes or phases in software development process and shortlist the various control factors at each phase. Preferably choose the parameters which the project manager can change without much difficulty by altering the team composition as control factors.

Step 3: Collect data on control factors, phase wise and delivered defect densities from the projects in the group.

Step 4: Develop models for phase wise defect densities in terms of control factors using a suitable modeling technique.

Step 5: Predict the phase wise defect densities using the respective models and compute the residuals

Step 6: Develop a model for delivered defect density in terms of predicted phase wise defect densities.

Step 7: Construct suitable control charts to monitor the residuals of the models.

Step 8: To apply the methodology to a new project, estimate the optimum phase wise defect densities to achieve the delivered defect density goal using the model developed for predicting delivered defect density. The strength and weakness of the project team can be considered as constraints while estimating the optimum phase wise defect densities.

Step 9: Identify the team composition which would result in the optimum values of control factors needed for achieving the required phase wise defect densities using models developed for predicting stage wise defect densities.

Step 10: Execute the project and at the end of each phase compare the actual defect density with the predicted defect density and plot the residuals in the respective control charts. Whenever the chart indicates out of control, carry out root cause analysis and take necessary actions. If required recalibrate the models

The application of the methodology is demonstrated using a case study in the next session

4. Case Study

This study is carried out for telecom domain projects. The critical sub processes or phases identified for study are design, coding and link testing. Through discussions with the project managers and software engineers, the control factors at every stage are identified. The list of the phase wise control factors is given in Table 1.

Table 1
List of control factors at different phases

Phase	Factor	Data Type
Design	Review Type	Categorical (Peer Review & Fagan Review)
	Review Coverage	Numeric
	Reviewer Skill	Categorical (Learned, Practiced & Expert)
	Domain Skill	Categorical (Learned, Practiced & Expert)
Coding	Review Type	Categorical (Peer Review & Fagan Review)
	Review Coverage	Numeric
	Reviewer Skill	Categorical (Learned, Practiced & Expert)
	Domain Skill	Categorical (Learned, Practiced & Expert)
Link Testing	Test Setup	Categorical (Average, Good & Very Good)
	Data Quality	Categorical (Average, Good & Very Good)
	Team Skill	Categorical (Learned, Practiced & Expert)
	Test Cases	Numeric
	Test Coverage	Numeric

The data on the control factors and phase wise defect densities are collected from the past projects in the telecom domain group and models are developed for predicting the phase wise defect densities. Since some of the factors are categorical and the remaining are numeric, models are developed using classification and regression tree algorithm (Myatt, 2007; Crawley, 2007) using R package (2016). The models are cross-validated at different sizes of the tree. The plots of cost complexity factor (cp) versus the cross-validation error (x-val Relative error) are given in Fig. 1 to Fig. 3.

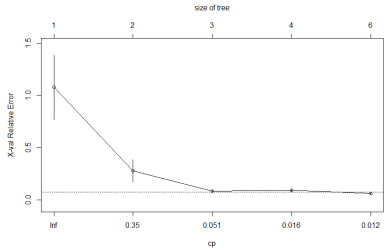

Fig. 1. cp versus cross-validation error plot for design review defect density (DR DD) model

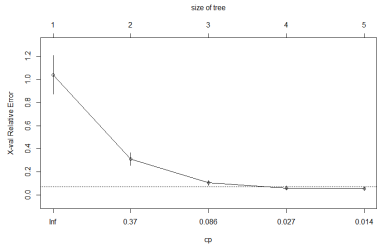

Fig. 2. cp versus cross-validation error plot for code review defect density (CR DD) model

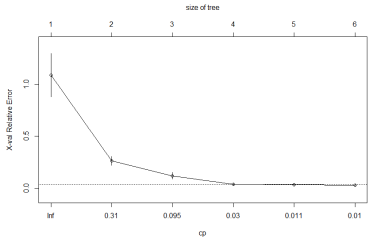

Fig. 3. cp versus cross-validation error plot for link testing defect density (LT DD) model

The best models are obtained by pruning the trees with cp corresponding to minimum cross-validation error (James et al., 2013). The best models obtained are given in Fig. 4 to Fig. 6.

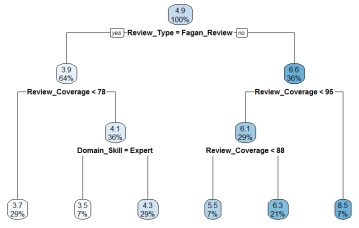

Fig. 4. Regression tree model for design review defect density

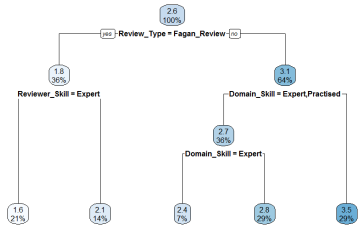

Fig. 5. Regression tree model for code review defect density

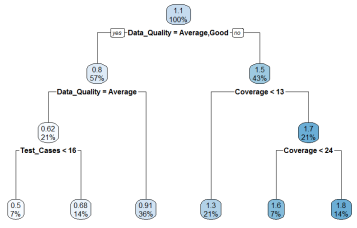

Fig. 6. Regression tree model for link testing defect density

The model diagnostic measures namely mean square error (MSE) and root mean square error (RMSE) of the models are given in Table 2.

Table 2
MSE and RMSE values of the models

Model	Design Review Defect Density	Code Review Defect Density	Link Testing Defect Density
MSE	0.0474	0.011	0.00073
RMSE	0.2177	0.105	0.0272

Table 2 shows that the RMSE values are reasonably close to zero. The residuals of the models are subjected to normality test. The Shapiro-Wilk normality test results are given in Table 3.

Table 3
Normality test results

Model	Design Review Defect Density	Code Review Defect Density	Link Testing Defect Density
Statistic	0.9483	0.9474	0.94921
P value	0.1795	0.1704	0.1892

Table 3 shows that the residuals of all the three models are normally distributed (p-value ≥ 0.05). Hence the residuals can be monitored using a control chart (Jayathavaj & Pongpullponsak, 2014; Black et al., 2011).

Finally, a model is developed for the delivered defect density in terms of predicted phase wise defect densities. The details of model development are as follows: The correlation matrix of the variables is given in table 4.

Table 4
Correlation matrix

	DRDD	CRDD	LTDD	Delivered DD
DRDD	1.00	-0.85	0.79	-0.80
CRDD	-0.85	1.00	-0.95	0.90
LTDD	0.79	-0.95	1.00	-0.86
Delivered DD	-0.80	0.90	-0.86	1.00

Table 4 shows that there is a good correlation between dependent variable delivered defect density and the predictors. But the correlation between predictors is also very high. Hence whether multicollinearity problem exists or not is verified by computing the variance inflation factor (VIF). The VIF values are given in Table 5.

Table 5
Variance Inflation Factor values

Predictor Variable	VIF Value
DR_DD	3.764322
CR_DD	15.52105
LT_DD	11.07212

Table 5 shows that the VIF value > 5 for two of the predictors. Hence model cannot be developed using ordinary least square regression. The multicollinearity can be tackled by dropping some of the correlated predictor variables or using principal component regression, partial least square regression, ridge regression, etc. Since all the important phase wise defect densities are important, dropping some of the predictor variables is not a good option in this scenario. The principal component and partial least square regression will first generate uncorrelated components which are the linear combinations of the predictor variables. Then the model is developed using these components as predictors. Even though no predictor is dropped in these approaches, the predictors would not be directly used in the model. Another option to tackle multicollinearity issue is to develop the model using ridge regression. The

ridge regression is a shrinkage methodology, which would give a simple model for dependant variable in terms of predictors but the coefficients of some of the correlated predictor variables are shrunk close to zero (Friedman et al., 2001). Hence it is decided to use ridge regression to develop the model using R package. The best value of the shrinkage parameter λ is obtained through cross-validation. The mean square error versus log (λ) plot is given in Fig. 7.

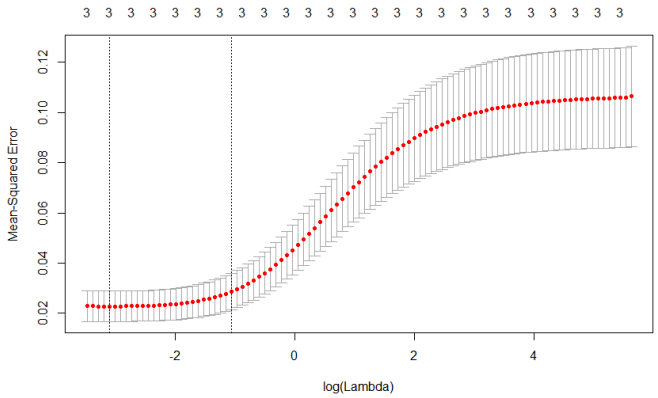

Fig. 7. MSE versus log (λ) of ridge regression model

The best value of λ and log (λ) obtained from Fig. 7 is given in Table 6.

Table 6
Best λ value

Log(λ)	-3.1105
λ	0.044578

The model coefficient obtained with best λ value using ridge regression is given in Table 7 and the model performance measures are given in Table 8

Table 7
Model coefficients

Variable	Value
Intercept	0.52685
DR_DD	-0.04589
CR_DD	0.17866
LT_DD	-0.21211

Table 8
Model performance measures

Statistics	Value
R^2	0.805
Adjusted R^2	0.78
MSE	0.0189
RMSE	0.1373

Table 8 shows that R^2 and adjusted R^2 are greater than 0.6. Hence the model is reasonably good. The normality of the model residuals is checked through Shapiro-Wilk test and normal quantile - quantile (Q_Q) plot. The Q - Q plot is given in Fig. 8 and Shapiro test results are given in Table 9.

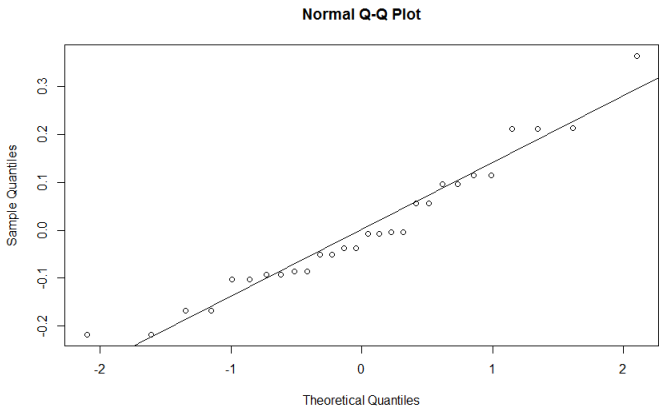

Fig. 8. Normal Q - Q plot of residuals

Table 9
Shapiro-Wilk normality test results

Statistic	Value
w	0.95325
p-value	0.2389

Table 9 shows that the p-value > 0.05 and also the points in figure 8 are more or less on a straight line. Hence the residuals of the ridge regression model for predicting delivered defect density also normally distributed. Since the residuals of all the models (table 3 and 9) are normally distributed, individual *x* control charts are constructed for monitoring the residuals and detecting the out of control situations (Bag et al., 2012; Noghondarian & Ghobadi, 2012). The control limits of the individual *x* control charts are given in table 10.

Table 10
Control limits of charts constructed to monitor model residuals

Variable	LCL	CL	UCL
DRDD Model residuals	-0.680	0.00	0.680
CRDD Model residuals	-0.3567	0.0000	0.3567
LTDD Model residuals	-0.0917	0.0000	0.0917
DeliveredDD Model residuals	-0.3978	0.0000	0.3978

For ease of implementation of the proposed methodology, a Microsoft excel macro based template is created. The screenshot of the template is given in Fig. 9.

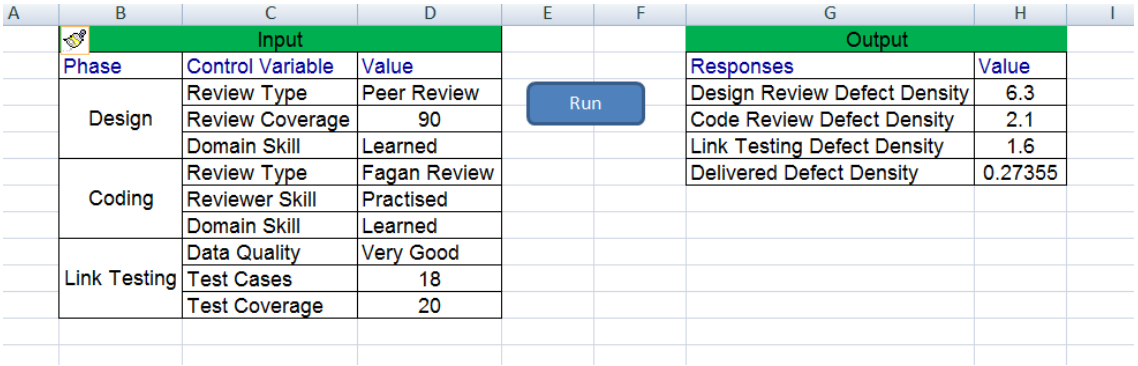

Fig. 9. MS Excel template for the implementation

The project managers or leaders can key in the values of control variables to the excel template (figure 9) and execute the macro by clicking on the run button. The template would display the predicted phase wise and delivered defect densities. If the delivered defect density is not close to the goal or target, then the managers can adjust the control variables in one or more phases and identify a feasible combination of control variable values which would give desired delivered defect density. Then execute the project with the feasible setting of the control variables. At the end of each phase, measure the actual defect density, compute the residuals and plot the residuals on the respective control charts. Whenever the control chart is showing out of control situation, carry out root cause analysis and take necessary action. If necessary, the model can be recalibrated.

The model is validated on seven projects which were not used for developing the models. The value of the control variables of the aforementioned projects is given in Table 11.

Table 11
Control variable values used for validation

Phase	Project id	1	2	3	4	5	6	7
Design	Review Type	Fagan Review	Peer Review	Fagan Review	Fagan Review	Peer Review	Fagan Review	Peer Review
	Review Coverage	80	70	90	75	85	85	100
	Domain Skill	Expert	Learned	Practised	Expert	Practised	Expert	Learned
Coding	Review Type	Peer Review	Peer Review	Fagan Review	Peer Review	Fagan Review	Peer Review	Fagan Review
	Reviewer Skill	Practised	Expert	Practised	Learned	Practised	Practised	Expert
	Domain Skill	Practised	Practised	Learner	Practised	Practised	Learner	Expert
Link Testing	Data Quality	Good	Good	Good	Average	Good	Average	Very Good
	Test Cases	80	70	110	40	60	8	90
	Test Coverage	8	12	11	5.5	7.7	4	15

The phase-wise predicted and actual defect densities along with the actual and predicted delivered defect density is given in Table 12.

Table 12
Validation results

	DR Defect Density		CR Defect Density		LT Defect Density		Delivered Defect Density	
Project Id	Predicted	Actual	Predicted	Actual	Predicted	Actual	Predicted	Actual
1	3.50	3.30	2.80	2.90	0.91	0.91	0.67	0.5
2	5.50	5.70	2.80	2.90	0.91	0.90	0.58	0.52
3	4.30	4.20	2.10	1.90	0.91	0.94	0.51	0.67
4	3.70	3.80	2.80	2.60	0.68	0.67	0.71	0.77
5	5.50	5.50	2.10	2.10	0.91	0.91	0.46	0.39
6	3.50	3.70	3.50	3.50	0.50	0.46	0.89	0.94
7	8.50	8.10	1.60	1.70	1.60	1.58	0.08	0.23

Table 12 shows that the defect densities predicted using the methodology is reasonably close to the actual defect density values. This showed that the proposed methodology can be successfully used for controlling the quality of the software development process.

The methodology has been pilot implemented on three projects. The values of the control factors, predicted defect densities using the macro tool and the target set on delivered defect densities are given in table 13. The table 13 shows that predicted delivered defect density is reasonably close to the target set for projects 1 & 3. Hence the projects 1 and 3 have been executed with the given setting. For the project 2, the predicted delivered defect density was 0.789 against a target of 0.6. Hence the project manager decided to change the control factors slightly and with the help of macro tool, identified that increasing the design review coverage from 60% to 80%, changing the development team composition such that the domain skill would change to “Practised” from “Learner” category, and increasing link testing test cases to 100 from 90 would give a predicted delivered defect density of 0.637 which was close to the

set target of 0.6 for the project. Hence the project 2 was executed with the changed settings. The changed values of control factors and expected defect densities are also given table 13.

Table 13
Implementation Data

Variables	Project 1	Project 2		Project 3
		Initial	Modified	
Review Type	Peer Review	Fagan Review	Fagan Review	Fagan Review
Review Coverage	80	60	80	60
Domain Skill	Practiced	Practiced	Practiced	Practiced
Review Type	Peer Review	Peer Review	Peer Review	Peer Review
Reviewer Skill	Learned	Expert	Expert	Expert
Domain Skill	Practiced	Learned	Practiced	Learned
Data Quality	Good	Good	Good	Average
Test Cases	90	90	100	90
Test Coverage	15	10	15	10
Design Review Defect Density	5.5	3.700	4.300	3.700
Code Review Defect Density	2.8000	3.500	2.800	3.500
Link Testing Defect Density	0.9100	0.910	0.910	0.680
Delivered Defect Density	0.5817	0.789	0.637	0.838
Target	0.5000	0.600	0.600	0.800

The actual defect densities measured after the execution of projects and the corresponding residuals are given in Table 14. Table 14 shows that the actual values are very close to the predicted values and the residuals are within the control limits of the respective control charts. Thus the pilot implementation has once again confirmed that the methodology can be used for software quality management and achieve the target set on delivered defect density.

Table 14
Implementation results

		Project 1	Project 2	Project 3
Design Review Defect Density	Actual	5.7	4.6	4
	Predicted	5.5	4.300	3.700
	Residuals	0.200	0.300	0.300
Code Review Defect Density	Actual	2.7	2.5	3.8
	Predicted	2.8000	2.800	3.500
	Residuals	-0.100	-0.300	0.300
Link Testing Defect Density	Actual	0.94	0.9	0.72
	Predicted	0.9100	0.910	0.680
	Residuals	0.030	-0.010	0.040
Delivered Defect Density	Actual	0.57	0.6	0.79
	Predicted	0.5817	0.637	0.838
	Residuals	-0.012	-0.037	-0.048

5. Conclusion

Quality, along with cost and schedule are important for the software firms to retain the customers as well as to get new projects from the customers. The software quality is generally expressed in terms of delivered defect density. The project managers need to quantitatively manage the software development process to achieve the goal on delivered defect density. The software development process consists of interlinked multiple phases or stages with the defect density at each stage impacting that at subsequent stages and the delivered defect density. The process performance models available for quantitative project management often model delivered defect density in terms of factors from different phases together or model defect density of a particular phase only. The multistage process control techniques are more suitable for quantitatively managing processes with multiple stages. In this paper, the authors suggested multistage statistical process control methodology for monitoring and controlling the software development process. The proposed methodology is a combination of process performance models and control charts.

The case study on the application of the suggested methodology for controlling projects of telecom domain is also discussed in the paper. The design, coding and link testing phases are identified as multiple stages in the development process. Using the data collected from past projects, models are developed for phase wise defect densities namely code review defect density, design review defect density and link testing defect density. The predictor variables are identified from the respective phases. Since the predictors are a combination of numeric and categorical variables, the models are developed using classification and regression tree technique. Then a model is developed for delivered defect density in terms of predicted phase wise defect densities as predictor variables. Since the phase wise defect densities are correlated and multicollinearity issue existed, the model is developed using ridge regression technique. Finally, control charts are developed to monitor the residuals of the models. An Excel macro based template is developed for implementing the methodology. The project managers can enter the values of the phase wise predictor variables in the excel template and run the macro. The macro will compute the phase wise and delivered defect densities using the model. The project managers can compare the predicted delivered defect density with the target and if the predicted delivered defect density is not close to the required target, the managers can use the macro template to identify the optimum combination of predictor variables which would bring the delivered defect density close to the target. The methodology is validated on seven projects. The methodology is pilot implemented on 3 projects and the results are very encouraging.

The main advantage of the proposed methodology is that it gives the project managers the flexibility to tighten or relax the control at different phases and still achieve the goal or target on delivered defect density. Even though the case study is from telecom domain, the same can be used to monitor and control project of any domain. Similarly, the methodology can be used for projects with any number of multiple stages or sub processes.

References

Adam Jr, E. E., Flores, B. E., & 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), 43-63.

Antony, J., & Fergusson, C. (2004). Six Sigma in the software industry: results from a pilot study. *Managerial Auditing Journal*, *19*(8), 1025-1032.

Asher, M., & Kanji, G. K. (1996). *100 Methods for Total Quality Management*. Sage Publications.

Bag, M., Gauri, S., & Chakraborty, S. (2012). Feature-based decision rules for control charts pattern recognition: A comparison between CART and QUEST algorithm. *International Journal of Industrial Engineering Computations*, *3*(2), 199-210.

Black, G., Smith, J., & Wells, S. (2011). The impact of Weibull data and autocorrelation on the performance of the Shewhart and exponentially weighted moving average control charts. *International Journal of Industrial Engineering Computations*, *2*(3), 575-582.

Ceylan, E., Kutlubay, F. O., & Bener, A. B. (2006, August). Software defect identification using machine learning techniques. In *Software Engineering and Advanced Applications, 2006. SEAA'06. 32nd EUROMICRO Conference on* (pp. 240-247). IEEE.

Crawley, M. J. (2012). *The R book*. John Wiley & Sons.

Ding, Y., Shi, J., & Ceglarek, D. (2002, January). Diagnosability analysis of multi-station manufacturing processes. In *ASME 2002 International Mechanical Engineering Congress and Exposition* (pp. 475-484). American Society of Mechanical Engineers.

Djurdjanovic, D. R. A. G. A. N., & Ni, J. (2001). Linear state space modeling of dimensional machining errors. *Transactions-North American Manufacturing Research Institution of SME*, 541-548.

Fenton, N., & Bieman, J. (2014). *Software metrics: a rigorous and practical approach*. CRC Press.

Friedman, J., Hastie, T., & Tibshirani, R. (2001). *The elements of statistical learning* (Vol. 1). Springer, Berlin: Springer series in statistics.

Hao, Y., & Zhang, Y. F. (2011, May). Statistical prediction modeling for software development process performance. In *Communication Software and Networks (ICCSN), 2011 IEEE 3rd International Conference on* (pp. 703-706). IEEE.

Harter, D. E., Krishnan, M. S., & Slaughter, S. A. (2000). Effects of process maturity on quality, cycle time, and effort in software product development. *Management Science*, *46*(4), 451-466.

Hawkins, D. M. (1993). Regression adjustment for variables in multivariate quality control. *Journal of Quality Technology*, *25*(3), 170-182.

Herbsleb, J., Zubrow, D., Goldenson, D., Hayes, W., & Paulk, M. (1997). Software quality and the capability maturity model. *Communications of the ACM*, *40*(6), 30-40.

Iso, I., & Std, I. E. C. (2001). 9126 Software product evaluation–quality characteristics and guidelines for their use. *ISO/IEC Standard*, *9126*.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An introduction to statistical learning* (Vol. 6). New York: springer.

Jayathavaj, V., & Pongpullponsak, A. (2014). A simulation study on the performance of the sign test, Mann-Whitney test, Hodges-Lehmann estimator and control charts for Normal and Weibull data. *International Journal of Industrial Engineering Computations*, *5*(4), 561-574.

Jin, J., & Shi, J. (1999). State space modeling of sheet metal assembly for dimensional control. *Journal of Manufacturing Science and Engineering*, 121(4): 756-762.

John, B., & Kadadevarmath, R. (2015). A methodology for quantitatively managing the bug fixing process using Mahalanobis Taguchi system. *Management Science Letters*, 5(12), 1081-1090.

Montgomery, D. C. (2007). *Introduction to statistical quality control*. John Wiley & Sons.

Myatt, G. J. (2007). *Making sense of data: a practical guide to exploratory data analysis and data mining*. John Wiley & Sons.

Noghondarian, K., & Ghobadi, S. (2012). Developing a univariate approach to phase-I monitoring of fuzzy quality profiles. *International Journal of Industrial Engineering Computations*, 3(5), 829-842.

Neil, M. (1992). Multivariate assessment of software products. *Softw. Test., Verif. Reliab.*, *1*(4), 17-37.

Paulk, M. C. (1993). Comparing ISO 9001 and the capability maturity model for software. *Software Quality Journal*, *2*(4), 245-256.

Pressman, R. S. (2005). *Software engineering: a practitioner's approach*. Palgrave Macmillan.

Samson, D., & Terziovski, M. (1999). The relationship between total quality management practices and operational performance. *Journal of operations management*, *17*(4), 393-409.

Shu, L., & Tsung, F. (2003). On multistage statistical process control. *Journal of the Chinese Institute of Industrial Engineers*, *20*(1), 1-8.

Shu, L., Apley, D. W., & Tsung, F. (2002). Autocorrelated process monitoring using triggered cuscore charts. *Quality and Reliability Engineering International*, *18*(5), 411-421.

Shu, L., Tsung, F., & Kapur, K. C. (2004). Design of multiple cause-selecting charts for multistage processes with model uncertainty. *Quality Engineering*, *16*(3), 437-450.

Shu, L., Tsung, F., & Tsui, K. L. (2005). Effects of estimation errors on cause-selecting charts. *IIE transactions*, *37*(6), 559-567.

Song, Q., Shepperd, M., Cartwright, M., & Mair, C. (2006). Software defect association mining and defect correction effort prediction. *IEEE Transactions on Software Engineering*, *32*(2), 69-82.

Tamura, S. (2009). CMMI and TSP/PSP: Using TSP Data to Create Process Performance Models.

Team, R. C. (2014). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. 2013.

Tsung, F., Li, Y., & Jin, M. (2008). Statistical process control for multistage manufacturing and service operations: a review and some extensions. *International Journal of Services Operations and Informatics*, *3*(2), 191-204.

Turhan, B., & Bener, A. (2007, October). A multivariate analysis of static code attributes for defect prediction. In *Quality Software, 2007. QSIC'07. Seventh International Conference on* (pp. 231-237). IEEE.

Zantek, P. F., Wright, G. P., & Plante, R. D. (2006). A self-starting procedure for monitoring process quality in multistage manufacturing systems. *IIE Transactions*, *38*(4), 293-308.