Industrial Engineering & Management
ISSN: 2169-0316
Kanan et al., Ind Eng Manage 2017, 6:1
DOI: 10.4172/2169-0316.1000210
Research Article
OMICS International

Economic Advantages of Utilizing the Integrated Quality Software Development Model

Mohammad Kanan[1]*, Gamal Weheba[2] and Ramiz Assaf[1]

[1]University of Business and Technology, Jeddah, 2136, KSA
[2]Wichita State University, Wichita, USA

Abstract

New contemporary software development models try to tradeoff among the three major aspects of concern; Cost, time and meeting customer requirements. One of the recently introduced software development models is the Integrated Quality Software Development (IQSD). This model builds on the advantages of both the prototyping and waterfall models and eliminates their limitations. This research presents the development of a cost estimation function that quantifies the economic benefits of implementing the IQSD model. Numerical analysis indicated that the IQSD model outperforms traditional development models from an economic standpoint.

Keywords: Integrated quality software development; Waterfall model; Prototyping model; Software development lifecycle; Cost estimation

Introduction

Today, there is a huge demand for computerized and automated business. Software development companies must deliver and produce software applications that meet customer satisfaction. In addition, both customers and developers have a high concern for development cost and time to penetrate the market. Both conventional and contemporary software development models allow for a tradeoff between cost and risk of not meeting customer satisfaction [1].

In addition, focusing on clarifying and understanding customer requirements is very critical since customer expectations increase by time, and new technologies are becoming more advanced. Spending more for the needed design efforts up front can lead to a cost reduction at the end of the software development life cycle. In contrast, incomplete design efforts can increase the cost of maintenance, as shown in Figure 1 [2].

The Waterfall Development Model has the advantage of low cost and less time, if and only if customer requirements are completely understood and clear [3,4]. The Prototyping Development Model has a good application, which involves customers in the development process, but this model has no obvious end; in other words, it is an open-ended process needing a larger budget and more time [5,6].

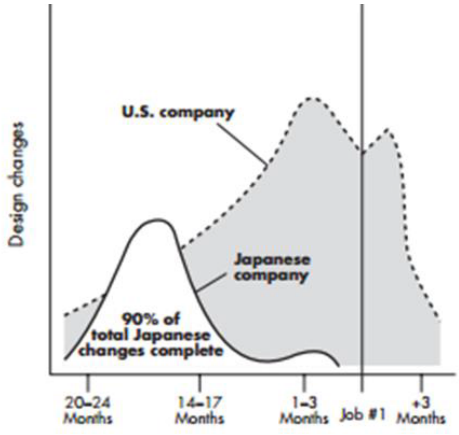

Figure 1: Design changes [2].

Integrated Quality Software Development (IQSD)

The only way to obtain customer satisfaction with low effort is by integrating a model that has the advantages of both the Waterfall Development Model and the Prototyping Development Model-combining these two models and using the advantages of clarifying the voice of the customer as shown in Figure 2. Once the voice of the customer is clearly understood, developers can then switch to the Waterfall Model, using its speed to complete the development process, thus meeting customer requirements and achieving quality.

Customer requirements/analysis

Understanding customer requirements and needs is the core and pillar of any successful process. To produce a successful software application, developers need to comprehend and understand all customers' voices clearly, since the final output or goal depends on their wants and desires in order to launch a successful product.

The first step in producing a successful software application is to define the problem to be solved and then define the intended customers. Customer requirements are derived from either customers or developers [7]. Customer requirements involve communication among these entities and can be categorized into functional and non-functional. Functional requirements are a subset of the entire application requirements and describe how the application or the system will work. Non-functional requirements explain the behavior of the application. In addition, there are many techniques used for collecting customer requirements:

One-on-one interviews: This most common technique focuses on sitting down with customers and inquiring about their needs, in other words, a direct interview between customers and developers, to avoid any misunderstanding of customer requirements [8].

***Corresponding author:** Kanan M, University of Business and Technology, Jeddah, and 2136, KSA, Tel: 966122159000; E-mail: m.kanan@ubt.edu.sa

Received February 17, 2017; **Accepted** February 24, 2017; **Published** February 28, 2017

Citation: Kanan M, Weheba G, Assaf R (2017) Economic Advantages of Utilizing the Integrated Quality Software Development Model. Ind Eng Manage 6: 210. doi:10.4172/2169-0316.1000210

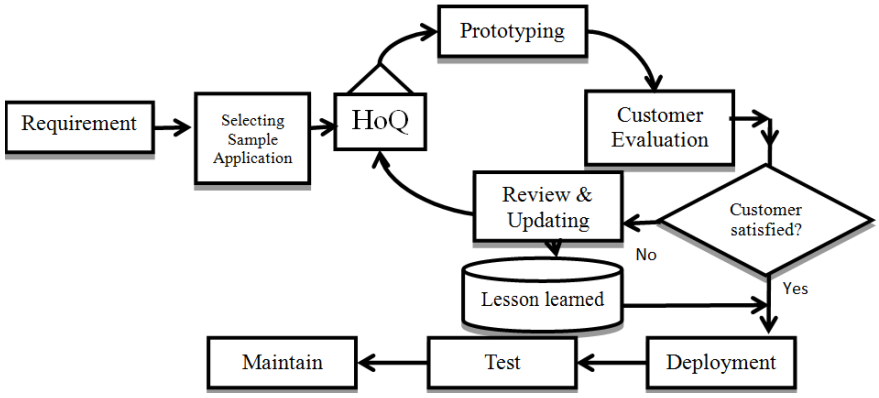

Figure 2: IQSD model [1].

Questionnaires: This technique is used when remote customers or a very large number of customers are involved, or there is no way to meet customers face to face. This technique must focus on avoiding redundancy in the large amount of data that is required [9].

Brainstorming: This technique is used when requirements are ambiguous and there is a need for innovative ideas [10]. First, developers are asked to meet in a room, start innovative brainstorming to solve a problem, and then find alternative solutions. Next, developers prioritize these alternatives. Finally, there is consensus as to the best alternative to finding an optimal solution.

Selecting the sample application

Selecting the sample application is very important in determining and judging the throwaway prototype. By sampling, developers can minimize and limit the possible liability of launching a "sub-par" product. Furthermore, bugs and defects during sampling can be fixed with minimum cost and time.

In order to select the sample application for iterative development, the system (application) must be dividable into subsystems; after that, a Pareto analysis technique for prioritizing these subsystems (subapplications) can be applied. Pareto analysis was discovered by the Italian scholar Vilfredo Pareto and is based on the Pareto principle where 80% of projects or problems are the result of 20% of causes.

In the stage of gathering requirements, customers should first determine the most needed subsystems to be developed and delivered, and then arrange them in ascending order. Based on this process, developers can prioritize subsystems by using Pareto analysis to arrange them according to the magnitude of their needs.

Designing prototype for selected application

Prototyping is a tool that explains whether requirements are met or not. In the prototyping phase, there are several steps, beginning with the house of quality.

House of quality: Sometimes, customers are not aware of exactly what they need, or their requirements are ambiguous. The HoQ technique looks for spoken customer requirements, thus making invisible requirements visible. This method is capable of capturing any misunderstanding of customer requirements by using a correlation matrix between what customers require and how developers design and engineer characteristics in order to meet customer satisfaction.

After gathering customer requirements, the HoQ can be used to translate all customer requirements into engineering characteristics to generate a set of features and functions to achieve customer satisfaction. By using the HoQ in software development, the quality of software will be increased and improved [11-14]. As shown in Figure 3, a series of steps is involved in the construction of the HoQ [13].

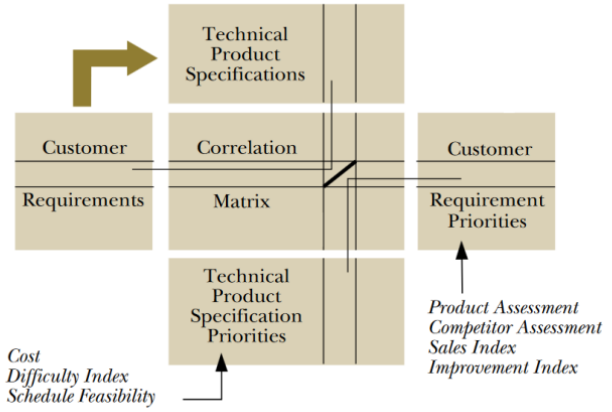

Figure 3: House of quality [13].

Developing a prototype: Early defect detection is recommended in order for developers to correct and fix any problems before system release. In addition, early defect detection can minimize the cost of poor quality. Prototyping enables customers to be involved during the design phase, so that customers can obtain a clear view and awareness of their requirements, which in turn will allow them to better share their ideas. All prioritized engineering characteristics in the HoQ will be implemented in the initial prototype.

Customer evaluation: By having a throwaway prototype, customers are ready to evaluate and provide feedback in order for changes to be made. Customer evaluation can help to implement an effective output system. Once this has succeeded, the next step is to deploy the entire development process for the remaining system subapplication. However, if the customer evaluation is not successful, then reviewing and updating is necessary.

Reviewing and updating: By using customer feedback, requirements and specifications can be improved. In addition, all lessons learned will be documented by using database storage to comprehend and document all customer feedback to incorporate the advantages and eliminate defects in order to accelerate the

development process. By using the lessons learned, this model can implement customer requirements for other subapplications during the prototyping phase, which runs the risk of not meeting customer requirements. Furthermore, this process saves time and cost.

Deployment

During this stage, all programming codes will be accomplished and implemented for the remaining subapplication in order to complete the entire application.

Testing

After accomplishing all required programming codes, they can be tested to ensure that neither bugs nor defects are found in matching customer requirements.

Maintenance

In this phase, the application will be ready to be released, and customers will be encouraged to send their feedback and comments relative to the correction of bugs or further improvement.

Software Cost Estimation

Due to the enormous growing demand for software applications, an appropriate method for cost estimation is needed. This method should be accurate and precise. There are two categories of software cost-estimation models: parametric and non-parametric. The parametric model comes from the statistical analysis of existent data, and the non-parametric model comes from expert and neural network methods. Many examples of parametric methods have been used in the software industry to estimate development cost; this research focuses on the Application Composition Model of the Constructive Cost Model II (COCOMO II) created by Boehm [15]

Application composition model

This model is based on the number of application points (i.e., screens, reports, and 3GL components) [16] and supports a prototyping-based project. To estimate the cost, the following steps are followed [16]:

1. Calculate object counts by estimating the number of screens, reports, and 3GL.

2. Categorize objects into three levels of complexity-simple, medium, and difficult-as shown in Table 1, where S, M, and D stand for simple, medium, and difficult, respectively.

3. Based on Table 2, provide a complexity weight for the number of each cell.

4. By adding the weighted objects, count the object points as one number.

5. Estimate the proportion of reused code; then use Equation 1 to calculate the new object point (NOP):

$$\mathbf{NOP} \quad \frac{(\mathbf{Object\ Points}) \times (100 - \%\mathbf{reuse})}{100} \tag{1}$$

6. Calculate the productivity rate (PROD) using Table 3.

7. Finally, calculate the person-months (PM) effort by using Equation 2:

$$\mathrm{PM} = \frac{\mathrm{NOP} \times (1 - \%\mathrm{Reuse})}{\mathrm{PROD}} \tag{2}$$

Parametric models are the most popular technique. They easily allow for modifying input data, and refining and customizing formulas. On the other hand, these models are unable to deal with different development environments. Furthermore, some experiences and factors cannot be quantified by using these models. Periodic calibration using a company's own data is required to assure accuracy [17].

Cost Estimation Function for the IQSD

This section proposes a new software cost estimation function to estimate the total efforts of the integrated quality software development model.

Notation

The following symbols are used in estimating the cost of utilizing the proposed IQSD model:

Y - Expected level of effort, in person-months, under the application composition model of COCOMO II

Y_1 - Cost of iterative development in PM

Y_2 - Cost of linear development in PM

a - Learning exponent

X - Expected number of iterations

g - Realization factor

p - Proportion of the sample application

Estimated cost of utilizing the proposed model

The IQSD model is aimed at reducing the risk of not meeting customer requirements and expectations. This is especially useful in developing customized (made-to-order) software systems. To determine the economic consequence of achieving this goal, a cost function for estimating the level of effort is proposed in Equation 3. This function accounts for two terms: one for the level of effort Y_1 expected during iterative development of a selected proportion of the system, and the other for the average effort Y_2 used during linear development of the reminder of the system. Both terms are estimated based on the expected level of effort Y obtained using the application composition submodel of the COCOMO II. This method is typically utilized to estimate the cost of employing the waterfall development model under the assumption of clear and fixed requirements. As such,

Number of Views	Screens			Number of Sections	Reports		
	Number and Source of Data Tables				Number and Source of Data Tables		
	Total <4 <2 serv <3 client	Total <8 2–3 serv 3–5 client	Total 8+ > 3 serv > 5 client		Total <4 <2 serv <3 client	Total <8 2–3 serv 3–5 client	Total 8+ > 3 serv > 5 client
<3	S	S	M	0, 1	S	S	M
3-7	S	M	D	2, 3	S	M	D
> 8	M	D	D	4 +	M	D	D

Table 1: COCOMO II Object Point Levels [16].

Object Type	Complexity-Weight		
	S	M	D
Screen	1	2	3
Report	2	5	8
3GL Component			10

Table 2: COCOMO II complexity weight of object points [16].

Developers' Experience and Capability ICASE Maturity and Capability	Very Low	Low	Nominal	High	Very High
PROD	4	7	13	25	50

Table 3: COCOMO II productivity [16].

the level of effort Y is considered a baseline estimate of the development cost expressed as

$$PM = Y_1 + Y_2 \quad (3)$$

Due to the advantages of the COCOMO II, as noted in section 3.1, it is assumed that potential users are familiar with the application composition model and have had more than one chance to calibrate its parameters.

Cost of iterative development: The term Y_1 accounts for the effort made during iterative development of a selected proportion p of the software system. This proportion is developed iteratively following the prototyping model. The resulting prototypes are used to clarify customer requirements and verify their capabilities. The proportion p is viewed as a representative sample of the system under development. It can be determined based on the ratio of its new object point to the estimated total NOPs of the software. The final result represents a functional component of the system that can be evaluated and accepted by the customer. Costs incurred during this iterative development depend on the selected proportion p and number of prototypes developed and evaluated, in an effort to clarify requirements. In obtaining an estimate of such costs, a production progress function is utilized to incorporate the effect of sequential learning on the cumulative cost. As frequently utilized in production planning and cost estimation, such a function requires an estimate of the cost of developing the first prototype p.**Y**, and the learning exponent α. The latter can be attributed to the gains expected from acquiring customer feedback during iterative development and the accumulation of lessons learned. Consequently, the expected effort of iterative development can be expressed as

$$Y_1 = p.Y.X^{1-\alpha} \quad (4)$$

As pointed out by Weheba and Elshennawy [18] it is common practice to estimate the exponent α in terms of the cost reduction for double production. Thus, each time the number of iterations is doubled, the cumulative average effort per iteration is expected to decrease by 2-α. Utilizing the level of effort from two successive iterations, an appropriate estimate of α can be obtained. It should be pointed out that the learning exponent in this application replaces the reuse rate in the COCOMO II, which is difficult to estimate a priori. The exponent represents a measure of competency of the software development team and its ability to translate customer requirements into technical specifications. With adequate training, higher values of α can be achieved.

The number of iterations X in Equation 4 is typically unknown due to the uncertainty involved. It is likely that the first prototype requires significant changes. And some changes may receive positive evaluations, while others may be shown to have no or even detrimental effect on customer satisfaction. The final prototype may differ considerably from that initially developed and evaluated. However, the number of iterations X can be assessed by using the realization factor g within the (0,1) interval as defined by Montgomery [19], who indicated that the number of iterations X can be approximated by a geometric random variable with an average 1/g. Here, the factor g represents the probability that the initial prototype will successfully achieve customer requirements. This is a function of the clarity of the initial requirements and past experience with the same customer. In general, it is appropriate to assume that projects with a high realization require less iterations on the average. In other words, development projects starting with clear and accurate requirements from returning customers should be assigned values of the factor g close to unity. Otherwise, initial subjective estimates of g may be used and updated as records accumulate. An initial value of < 0.25 is appropriate, as recommended by Yelle [20]. Also, it is important to note that Equation 4 indicates that the lower the realization factor g of the first prototype, the higher the effect of the exponent α on the estimated cost. This is expected to compensate for the effect of assuming a constant realization as a characteristic of the geometric distribution.

Cost of linear development: The cost of linear development Y_2 can be estimated using Equation 5, which accounts for the PM effort required for developing the remaining proportion 1–p of the system. It is assumed that the development will follow a linear model (waterfall) with a clear and accurate set of requirements. As shown in Figure 2, this stage is aided by lessons learned during iterative development. Such information is typically documented in the house of quality, with clear indications of user requirements and specific design aspects known to achieve them. It is assumed that this stage of development can begin only when a target level of customer satisfaction has been achieved. Utilizing the estimate Y from the COCOMO II, the linear development effort is expected to be

$$Y_2 = (1-p).Y \quad (5)$$

Total cost function: The total development effort in person-months of utilizing the proposed model can now be estimated by adding Equations 3 and 4:

$$PM = p.Y.X^{1-\alpha} + (1-p).Y$$
$$= Y\left[1 + p\left(X^{1-\alpha} - 1\right)\right] \quad (6)$$

An examination of Equation 6 reveals that the theoretical minimum level of PM effort can be achieved when p=0 (equivalently $X^{1-\alpha}=1$). At this level, the complete system is developed without iterations at the baseline level of PM effort. This entails the assumption of an accurate understanding of a fixed set of requirements as in the waterfall model. However, should this assumption be violated, then the actual effort of repeated development using the waterfall model is expected to be a multiple of Y, depending on the number of developments required to achieve customer satisfaction. On the other hand, when p=1, the development will follow the prototyping model at an estimated effort represented by a multiple (magnitude of $X^{1-\alpha}>1$) of the baseline development effort Y. The main advantage of the proposed IQSD model lies in its ability to represent developers with a middle ground approach, one in which the risk of failure is reduced at a fraction of the cost of repeated development.

The expected level of effort PM as a function of the development model utilized is shown in Table 4, for a development project with an initial effort Y of 5.0 PM (NOP/PROD=5.0). Values of PM for using the IQSD model were calculated based on Equation 6 at various levels of the realization factor g, assuming p=0.25 and α=0.40. For the waterfall

X	*g*	Efforts of Integrated Model	Efforts of Prototyping Model	Efforts of Waterfall Model (with iterations)
1.00	1.00	5.00	5.00	5.00
2.00	0.50	5.64	8.00	10.00
3.00	0.33	6.17	9.80	15.00
4.00	0.25	6.62	10.88	20.00
5.00	0.20	7.03	11.53	25.00
6.00	0.17	7.41	11.92	30.00
7.00	0.14	7.77	12.15	35.00

Table 4: Comparison of three models based on assumed data.

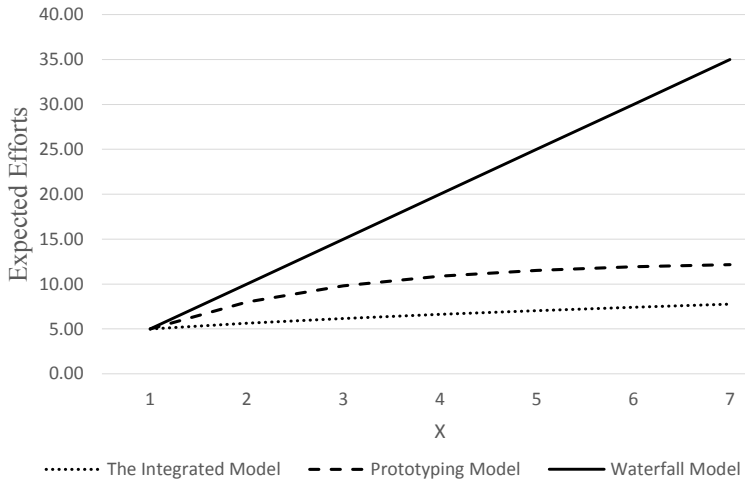

Figure 4: Comparison of PM effort based on development model used.

and prototyping models, values of PM were calculated based on the application composition submodel of the COCOMO II. The calculated value of the average PM when using the prototyping model is based on a reuse rate of 40%.

The value of the average PM for using the waterfall model, assuming a constant and fixed number of requirements, is calculated at the hypothetical level of realization (g=1.0). However, when this assumption is violated, values of g are used to calculate the expected number of redevelopments. This is typical of the waterfall model, as was noted in section 1.

Model performance

Calculated values of the average PM as a function of the development model used at the expected values of X (or 1/g) are represented graphically in Figure 4. As shown, all three models result in the same average level of PM when no iterations are needed. As the number of iterations increase, the IQSD model tends to outperform both the waterfall and prototyping models. Section 4.3 represents a study of the model performance at varying levels of its parameters.

Conclusions

The objective of this paper was to demonstrate the economic advantages of utilizing the IQSD model. In achieving this objective, a mathematical cost function for estimating the total effort (person-months) was developed. This cost function employs the level of effort obtained by using the application composition model of the COCOMO II as a baseline, given its popularity. It includes two terms: the first accounts for the level of effort required during the iterative development of a selected portion of the system, and the second accounts for the level of effort required for developing the remaining proportion. The total cost is estimated based on four factors: the expected baseline effort (Y) obtained using the COCOMO II, the proportion of the model selected for iterative development (p), the realization factor (g), and the learning rate (α). Numerical investigation of the model performance over practical levels of these four factors was conducted. The investigation utilized a two-level factorial arrangement and revealed that the expected total effort is more sensitive to changes in the realization factor (g) at the low levels of the learning rate (α). This indicates that high levels of learning are needed when developing software systems for new customers. It was noted that the model is not sensitive to changes in the proportion (p) selected for iterative development. This supports the effective utilization of the HoQ in translating customer requirements into engineering characteristics. In other words, users of the proposed development model should be more concerned with the ability of the selected proportion to reflect as much of the customer requirements, rather than its relative size [21-26].

The IQSD model could be used in product design where rapid prototyping and 3D printing are efficiently utilized for iterative development. This is an area where numerous research efforts have been made to reduce cost and time to market while improving design quality. The model is simple, easy to implement, and reinforces the need for clear communication between developers and customers. It allows developers to utilize customer feedback during the early stages of product development and achieve high levels of satisfaction.

References

1. Kanan M, Weheba G, Jaradat B (2014) Development of the Audit-Calc Software System: A Case Study. Journal of Management & Engineering Integration 7: 1.
2. Hauser JR, Clausing D (1988) The house of quality. Harvard business review 66: 3.

3. Amlani RD (2012) Advantages and limitations of different SDLC models. International Journal of Computer Applications and Information Technology 1: 6-11.
4. Sage AP, Palmer JD (1990) Software systems engineering. Wiley-Interscience.
5. Davis AM, Bersoff EH, Comer ER (1988) A strategy for comparing alternative software development life cycle models. IEEE Transactions on Software Engineering 14: 1453-1461.
6. Jalote P (2012) An integrated approach to software engineering. Springer Science & Business Media.
7. Maiden N (2008) User requirements and system requirements. IEEE Software 25: 90-91.
8. Gause DC, Weinberg GM (1989) Exploring requirements: quality before design. New York: Dorset House Pub, p: 299.
9. Foddy W (1994) Constructing questions for interviews and questionnaires: Theory and practice in social research. Cambridge : Cambridge university press, p: 228.
10. Osborn AF (1963) Applied Imagination; Principles and Procedures of Creative Problem-solving: Principles and Procedures of Creative Problem-solving. Scribner, p: 417.
11. Basili VR, Briand LC, Melo WL (1996) How reuse influences productivity in object-oriented systems. Communications of the ACM 39: 104-116.
12. Chavez A, Tornabene C, Wiederhold G (1998) Software component licensing issues: A primer. IEEE software 15: 47-53.
13. Haag S, Raja M, Schkade LL (1996) Quality function deployment usage in software development. Communications of the ACM 39: 41-49.
14. Poulin JS, Caruso JM, Hancock DR (1993) The business case for software reuse. IBM Systems Journal 32: 567-594.
15. Boehm B (1995) Cost models for future software life cycle processes: COCOMO 2.0. Annals of software engineering 1: 57-94.
16. Boehm B, Abts C, Clark B, Devnani-Chulani S (1997) COCOMO II model definition manual. The University of Southern California.
17. Lee S, Titchkosky L, Bowen S (2002) Software Cost Estimation. Department of Computer Science, University of Calgary.
18. Weheba GS, Elshennawy AK (2004) A revised model for the cost of quality. International Journal of Quality & Reliability Management 21: 291-308.
19. Montgomery DC (2008) Design and analysis of experiments. John Wiley & Sons.
20. Yelle LE (1979) The learning curve: Historical review and comprehensive survey. Decision Sciences 10: 302-328.
21. Weheba GS, Nickerson DM (2005) The Economic Design of -x Charts: A Proactive Approach. Quality and Reliability Engineering International 21: 91-104.
22. Sommerville I (2007) Software Engineering. Harlow: Pearson Education Limited. ISBN 978-0-321-313799.
23. Pandey P (2013) Analysis of the Techniques for Software Cost Estimation. In IEEE pp: 16-19.
24. Lee T, Choi D, Baik J (2010) Empirical study on enhancing the accuracy of software cost estimation model for defense software development project applications. in IEEE 2: 1117-1122.
25. Fine CH, Porteus EL (1989) Dynamic process improvement. Operations Research 37: 580-591.
26. Chung L, Sampaio JC (2009) On non-functional requirements in software engineering. in Conceptual modeling: Foundations and applications. Springer pp: 363-379.