SIMULATION RESULT: ENGAGE SOFTWARE TESTER EARLY IN SOFTWARE DEVELOPMENT LIFE CYCLE A PRIORITY BASED MODELS

[1]MANEESH V. DESHPANDE, [2]SURYAKANT B.THORAT, [3]PRADEEP K.BUTEY

[1]Asst. Professor RAICSIT, Wardha, [2]Director, ITM Nanded, [3]Coordinator MCA, Nagpur

Abstract— Many Research have been proposed, on engaging software tester early in software development life cycle. But very a few researchers have provided the guidelines to actually implement the same. This paper provides the complete environment and stated newly created models based on the priority assign to the testers in the software development. Considering normal SDLC, have some major issues like time, bug counting ratio any many more. But try to minimize the same complexities newly created models stated. In this paper Priority considered based on the experience point of view that the Tester actually have in the software company. Result of the simulation depends on comparison between normal SDLC and newly created models. Furthermore major consideration of comparison is on time required to complete all the phases of software development, number of bug count and average bug finding ratio.

Keywords— Software Development Life Cycle (SDLC), Software Testing (ST), Orange HRM Software, My Info module, Comma separated value files (CVS), bug, bug counting ratio.

I. INTRODUCTION

Development Process is essential in every change related process. Due to the same human being can able to portray the final product. If the development process is proper and implement in correct form then quickly can reach up to final stage without much more difficulty, but if not implemented properly or ignore some steps then can't fulfill the users requirements. The same case happened with software development and if software product should be proper incase of error free and user friendliness then software tester must act like an actor in film.In SDLC software Developer and software Tester are the basic building blocks. But in most companies these two are work independently. Because of their independent work many problems arises. But if they work in joint venture then this software development is called as Joint Application. A software development process, also known as a software development life cycle (SDLC), is a structure imposed on the development of a software product. It is often considered as a subset of system development life cycle. Software Testing is essential and very much crucial, not only for customers but for software companies also. Because of the awareness most software companies are using software testing. But some companies are still not implementing testing in proper stage. This paper tried to prove the same thing and mentioning some suggestions for engage software testers early in software development life cycle. Paper organized as follows our work in the context of prior work (Literature review), finding based on simulation result considering newly created models, and finally involves conclusion acknowledgement and references.

II. LITERATURE REVIEW

Mark L. Gillenson, PhD, CCP, October 16, 2006 [1]. The University of Memphis Research Proposal. "Engaging Testers Earlier in the SDLC", have the following views: We believe that software development is fertile ground for the use of cross-functional teams, with testers being integral members of those teams at all stages of development. An additional contribution will be testers seeing themselves as stakeholders in the quality of the finished applications by virtue of their work throughout the SDLC. This will lead to the further development of systems testing as a recognized and respected specialty within information systems organizations. Finding and fixing these problems early (i.e. at the requirements or design phase) will reduce the overall risk and cost of the product [2], this paper focuses on the software inspection approach. Next Paper based on the measuring the software quality during life cycle software development, with the help of improving the ISO 9126 [3]. Author describes the importance of testing throughout software development [4] and further believes that software testability analysis can play a crucial role in quantifying the likelihood that faults are not hiding after testing does not result in any failures for the current version. How software testing changes its nature in terms of working from organization to organization is stated in this paper. It is advisable to carry out the testing process from the initial stages, with regard to the Software Development Life Cycle or SDLC to avoid any complications [5]. Testing continues to represent the single largest cost associated with the development of sophisticated, software intensive, military systems. If the concept of testing begins very early in the development process significant savings can be achieved [6]. The focus of this paper is to present the first phase of development of decision models that could be used to determine the best uses of software testing resources. The ultimate model could be used to reduce overall costs while applying resources

where they provide the greatest value throughout the systems development process [7]

III. FINDINGS

Following are newly created models, for "Engage Software Tester Early in Software development Life Cycle." Considering Mark Gillenson (2007) models, newly created models are conceptually same but the difference is that tried to reach more in depth of the concept. In terms of mentioning Tester A and Tester B [1], categorized the testers on the basis of the experience i.e. senior tester and junior tester. All these models are implemented on each phases of traditional software development life cycle. Based on the work assign to individuals the concern models took birth as described below.

Model 1: Tester Centric Top Loaded Priority Assign Bifurcated Unbalanced Model:
The following figure concern with the software development. Since software development consider all the phases starting from user requirements, analysis where detailed study is done, design, coding, implementation post implementation and up to maintenance.

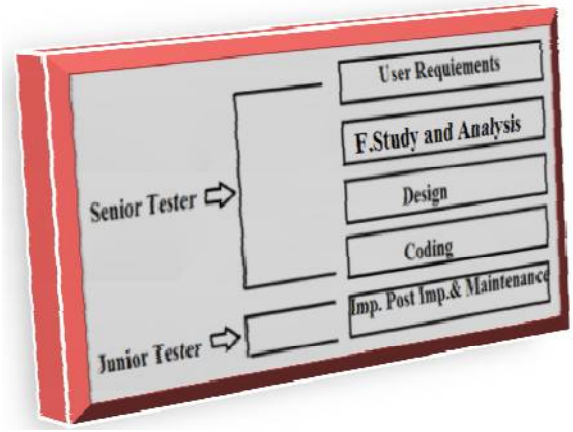


Figure 1-Tester Centric Top Loaded Priority Assign Bifurcated Unbalanced Model

But in all such concern phases specialist persons are used. The intention is to use software tester in above way as shown in figure. In above figure all the work stating from checking user requirements up to maintenance phase assign to software testers. The major terms i.e. phases are requirement up to coding is largely handled by senior tester, so that each phase becomes error free. Then comes to final stage of Implementation and Maintenance where we can able to assign the junior testers since not much be there in the final stage because of filtering in the form of testing previously in all the stages. The concern figure is called "Tester centric top loaded priority assign bifurcated unbalanced model" since in all phase software testers used. The reason for Top loaded priority bifurcated unbalanced assign to senior tester who has done most of the work and only small kind of work (Testing) assigned to junior tester.

Model 2: Tester Centric Bifurcated Balanced Model
The current model tries to minimize the work load. In above model the most of the work assigned to senior tester. So this model uses the combination of testers based on experience. In this model basic level testing is done by junior testers and detailed testing is done by senior once. This model named as "Balanced Model" since each phase of SDLC is equally shared by the combination of testers. This model also helps us to provide quality software. The following figure shows the same.

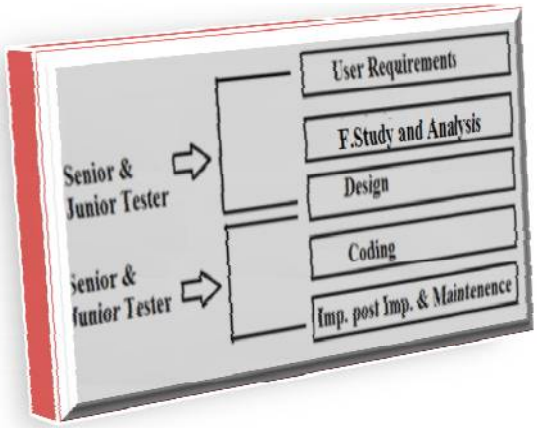


Figure 2-Tester Centric Bifurcated balanced Model

Model3: Combined / Mixed Priority Assign Bifurcated Balanced Model
Above two models are testers centric. In the software development all the members are very crucial and if anyone ignored, it directly affects the product. So the intension is to use combined / mixed priority in software development.

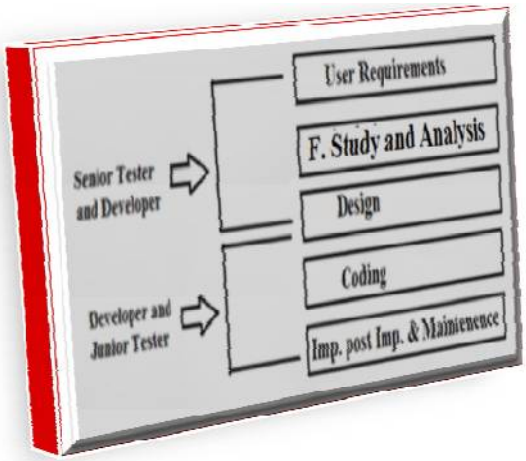


Figure 3- Combined / Mixed Priority Assign Bifurcated Balanced Model

The above figure is the "Combined/mixed Priority Assign Bifurcated Balanced Model", where upper half important work assigned to the Experience

Testers and Developers. The documentation format will be ended at very high note. The rest work is assign to Senior Developer and junior testers. In this model we used a different mentality personal so that the final product will be farbetter than traditional development model.

IV. SIMULATION RESULTS

For successful creation of simulator OrangeHRM Open Source HR Management software used. Wefocused OrangeHRM – My Info Module. The reason of choosing the above software is because it's easily available on internet, most of IT companies used the same software and we don't have to pay (Open Source). The technical details are as follows.

- ✓ OrangeHRM version 2.7
- ✓ Frontend- Java version 1.7
- ✓ Backend- excel , Comma Separated Values file (CVS)
- ✓ Ubantu- Operating System
- ✓ Bugzilla- version 4.4.9.

Simulation software based on the new created models, in which Software Testing field gives the prime importance. All the comparison based on

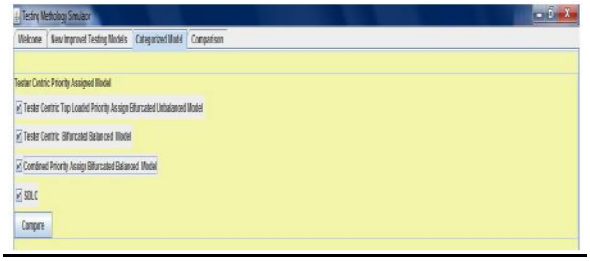
A) Duration: (Time required for completing all the phases of software development in hours)

B) Bug Count: (Total Quantity of bug found during software development)

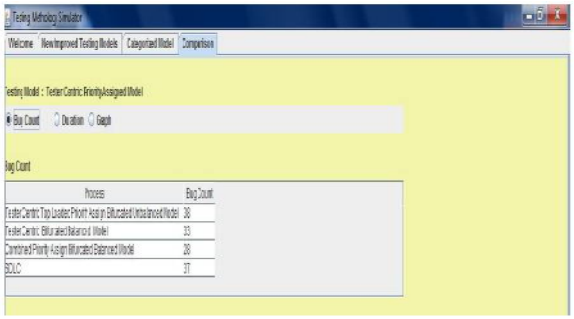
C) Graph : (Including Design Efforts, Execution Efforts and Total Efforts)

First "Welcome Page" Snapshot Related To Simulator, "For Engage Software Tester Early In Software Development Life Cycle".

Comparison On the basis of Priority Assign between Top Loaded Bifurcated Unbalanced, Bifurcated Balanced, Combined priority Bifurcated Balanced and Software Development Life Cycle Model: Now here comparison took place between all the above three newly created tester centric models. Following snapshot shows the same below.

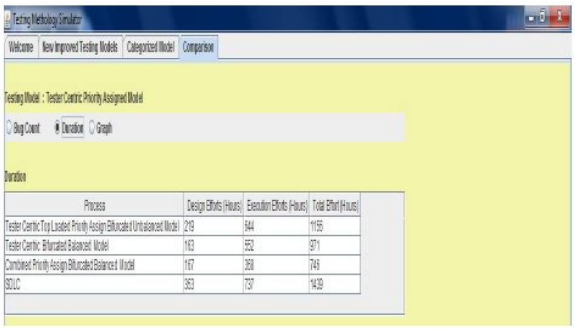


Bug Count: Following bug count snapshot appears with the comparison of four concern models.

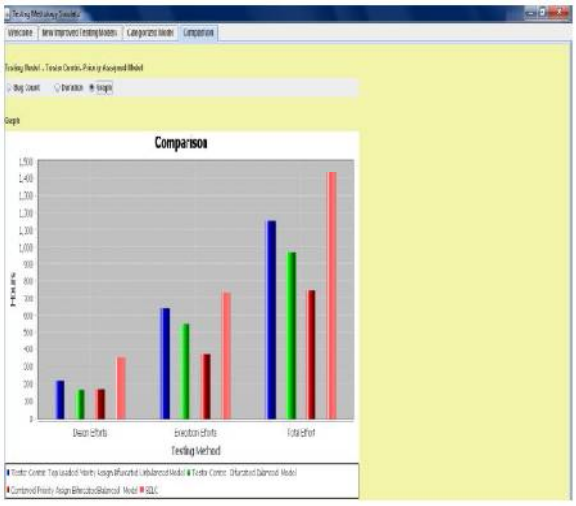


Duration: Following duration snapshot appears with the comparison of four concern models.

{Total Efforts= Design effort + Execution Effort + UAT + UAT Support + Contingency factor +Training to support team.}



Graph: Finally the graph which shows Design efforts, Execution efforts and Total efforts. A following graph consists of testing method on X-axis and Total hours required on Y-axis with the comparison of four concern models.



A) Comparison based on Total time required to complete software development:

Following table shows all important findings based on the simulation software result.

Table 1: Result of simulation based on time required.

Sr. N	Models Type	Required Time In Hrs.	Bug Count	Distribution Time in Hrs (Design+Execution+Other)	Bug Percentage ratio
1.	Normal **Software Development** Life Cycle	1439	37	353+737+349	2.57%
2.	Tester Centric **Top Loaded** Priority Assign Bifurcated Unbalanced Model	1156	38	219+644+293	3.28%
3.	Tester Centric **Bifurcated Balanced** Model	971	33	163+352+256	3.39%
4.	Tester Centric **Combined Priority** Assign Bifurcated Balanced Model	746	28	167+368+211	3.75%

B) Comparison based on Bug Count considering Stated Models: Following table shows total bug count during all the phases of software development. Starting with Normal Software development and comparison with all newly created models. Newly created models numbers have their usual meaning mentioned earlier.

Table 2: Result of simulation based on Bug Count

Normal SDLC Bug Count	New Stated Models Bug Count
37	1) 38 2) 33 3) 28

C) Comparison Based On percentage of Bug Count Considering Stated Models:
Next consideration is based on the bug finding ratio which is also shown in following table.
{Bug Finding Ratio= Bug Count/Required time in hours to complete all phases}

Table 3: result if simulation based on Average Bug Count Ratio

Normal SDLC Bug Finding Ratio	New Models Bug Finding Ratio	Average Bug Findings Ratio
2.57%	1) 3.28% 2) 3.39% 3)3.75%	3.47%

CONCLUSION

This paper suggests how to engage software tester early in software development life cycle. Software Product is big issue for customer as well as Software Companies. Because of the same software product should be delivered on time with budget and without much more errors. While implementing the above stated models certainly software companies required a less time as compared to normal software development as mentioned in following table 4.

Table 4: Showing required time in hours to complete software development phases

Sr. N	Models Type	Required Time In Hrs.
1.	Normal **Software Development** Life Cycle	1439
2.	Tester Centric **Top Loaded** Priority Assign Bifurcated Unbalanced Model	1156
3.	Tester Centric **Bifurcated Balanced** Model	971
4.	Tester Centric **Combined Priority** Assign Bifurcated Balanced Model	746

While considering the above table normal SDLC requires 1439time (in hours) to complete the process, but if stated models implemented with all considered suggestions, then the result is not only more positive but eye catching also in the form of 1156, 971, and 746. So time management is reduced in this case. Secondly bug counting is also essential, and tried to improved the same as normal SDLC as shown in following table.

Table 5: Showing Bug count and Findings.

Sr. N	Models Type	Required Time In Hrs.	Bug Count	Findings
1.	Normal **Software Development** Life Cycle	1439	37	0.025
2.	Tester Centric **Top Loaded** Priority Assign Bifurcated Unbalanced Model	1156	38	0.032
3.	Tester Centric **Bifurcated Balanced** Model	971	33	0.033
4.	Tester Centric **Combined Priority** Assign Bifurcated Balanced Model	746	28	0.037

So normal SDLC bug counting is 0.025, and all stated models much more improved as well in terms of bug counting also as 0.032, 0.033 and 0.037. Above table findings generated by simply dividing bug count with required time in hours. Lastly consider bug counting

ratio in terms of percentage mentioned in following table as well.

Table 6: Showing Improved model

Normal SDLC Bug Finding Ratio	New Models Bug Finding Ratio
2.57%	1) 3.28% 2) 3.39% 3) 3.75%

Out of the three suggested models the most Reliable and beneficial model is the one having the bug finding ratio is "3.75".Named as **"Combined / Mixed Priority Assign Bifurcated Balanced Model".**

ACKNOWLEDGEMENT

I would like to thank all anonymous reviewers for their valuable comments that were used to improve this paper. I am grateful to **Dr. Suryakant B. Thorat** and **Dr. Pradeep K. Butey** for giving proper guidelines and provide necessary help to me. Finally last but not the least my family, my all mentors starting from my childhood and my friends for their kind support.

REFERENCES

[1] "Engaging Testers Early and Throughout the SDLC includes seven model" By Mark L. Gillenson, Xihui Zhang, Sandra Richardson.
[2] Finding and Fixing Problems Early:A Perspective-Based Approach to Requirements and Design Inspections by Dr. Forrest Shull and Dr. Ioana Rus , Dr. Jeffrey C. Carver.
[3] Measuring the Software Product Quality during the Software Development Life-Cycle: An International Organization for Standardization Standards Perspective By Rafa E. Al-Qutais (Journal of Computer Science 5 (5): 392-397, 2009 ISSN 1549-3636 © 2009 Science Publications).
[4] Improving the Software Development Process Using Testability Research.
[5] Author: Jeffrey M. Voas Keith W. Miller.
[6] Importance of Testing in Software Development Life Cycle Author: T.Rajani Devi (International Journal of Scientific & Engineering Research Volume 3, Issue 5, May-2012 1 ISSN 2229-5518).
[7] Testing and Software Technical Risk Assessments Author: Brad Neal, Sim Ventions.
[8] Development of Decision Models for Best Use of Software Testing Resources Author: Charles J. Campbell, Judith C. Simon, Ronald B. Wilkes.

★★★