Simulation Result: Engage Software Tester Early in Software Development Life Cycle a Bug life Cycle Model

Dr. Maneesh V. Deshpande

Asst. Professor, Computer Science Department, S.S. Maniar College, Nagpur, Maharashtra, India

ABSTRACT

In this modern and developed era Software Testing still be most neglected phase in software development. Like oxygen is necessary for the human being, in the same way software testing is necessary for the software product. This paper focuses on the bug life cycle. Finally the complete environment Result of the simulation depends on normal SDLC and newly created model (Bug life Cycle). 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.

Key terms: Software Development Life Cycle (SDLC), Software Testing (ST), Orange HRM Software, My Info module, Comma separated value files (CVS), bug, bug counting ratio, bug life cycle.

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.[1]. 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. While talking about testing there are following most important terms are always consider.

* **Defect**- Is the difference between expected and actual output.
* **Mistakes**- A human action that produces an incorrect result.
* **Failure**- A incorrect result by which the system will crash.
* **Error or Bug**- Is any incorrect coding in the syntax of a program give rise to defect.

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 model (Bug Life Cycle), and finally involves conclusion acknowledgement and references.

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],This paper largely focuses on pitfall of software testing[8].

FINDINGS

Following model is newly created model (Bug Life Cycle), for “Engage Software Tester Early in Software development Life Cycle.”

❖ **Bug life Cycle:** Bug Life Cycle indicates the process starting with bug found to fix the same bug. In this bug life cycle after finding bug the bug is directly assign to the software developer for necessary correction. After proper monitoring and evaluating,developer sends the same to tester. i.e. the term called as “Reassigning” the work to tester. Then next stage if the tester is satisfy with the corrected code it must passed to the developer for further verification and if all goes well the fixing the bug process took place and responsible persons for fixing the bugs are TD (Team Director)/TL (Test Lead) developer.

But if the tester sees the same code or bug is not corrected properly then it must be assign to the developer for “Re-Correction” of bug. The same sequence must be followed for all generated bugs. The corresponding figure of traditional bug life cycle is as shown below.

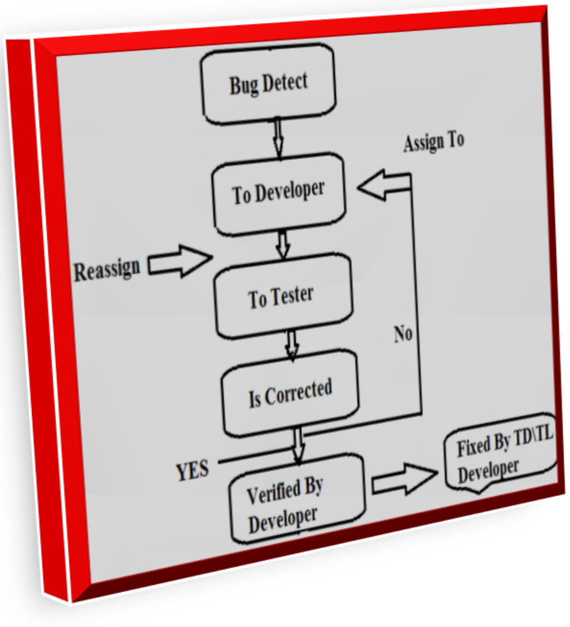


Figure 1 - Traditional Bug life cycle

While considering the traditional bug life cycle the following point need to be considered.

- ✓ The same bug is assigned to developer first.
- ✓ If the series of bugs are coming from customers then the quality of product is not proper.
- ✓ The tester is used at lateral stages of the bug life cycle.
- ✓ If the tester doesn’t know the same bug. Tester can’t be correct the bug.
- ✓ If proper correction needed then tester must provide proper information to the developer. In such case only detailed information is not sufficient but must provide the snap shot of infected portion.

- **New Improved Tester Centric Bug Life Cycle:** Following figure shows the new improved bug life cycle. Following changes has made to improve the quality of the product as well as the software development life cycle. Following are the number of possible ways from where we can able to achieve the bugs.

1) From Developer (White Box Testing)
2) From Tester (Black Box Testing)
3) lastly From User (User Acceptance Testing)

Identification of bugs generally received from developers, mostly from testers and very rarely from user. If we engage the software testers early in the software development stages then this problem of collecting the bugs from the user might be reduced and mostly founder of bugs are the testers. Following figure shows the new improved tester centric bug life cycle.

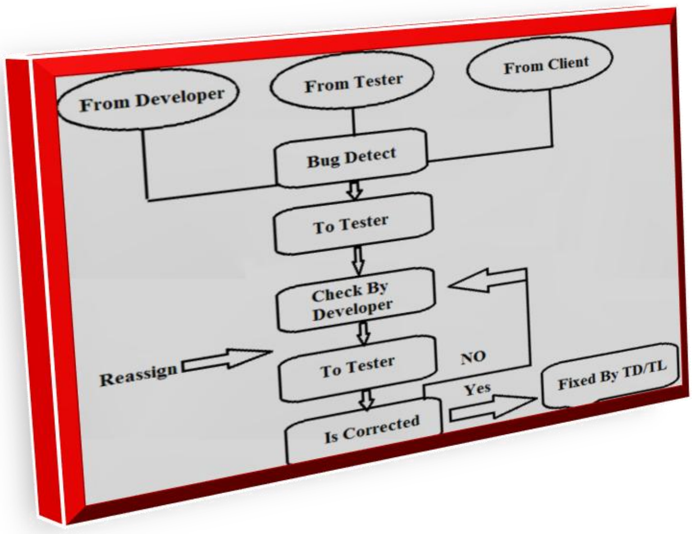


Figure 2: New Improved Tester Centric Bug life cycle

In traditional bug life cycle the ratio for finding bugs are in the range of 70%-30%. (70% bugs coming from the users after installing the software to user environments and 30% from the software testers.) but if the software companies agree to engage software testers early in software development then this ratio might be change and ratio somewhere in between 65%-35%.(65% bugs found by software testers while developing the software and remaining 35% coming from the users). Engaging software testers early in software development, quality of the software much improve and companies can able to reach the user satisfaction level as well.

In the above figure consider the three ways where we can able to find the bugs. After detecting the bugs it will directly assigned to software tester and not to the developer. In this case software tester seesthe same bug first. If the same bug is not proper (Valid) then software tester directly rejects the bug. If the bug is proper tester test the same bug before reporting to the software developer, same bug is handover to the software developer for necessary improvement. After correcting the bug it will reassign to the software tester. Once again the tester works on the same bug. If the same code is work properly, it will pass on to the team leader or team director for fixing otherwise it will again send to the software developer for correction and same procedure continue until the same code becomes proper.

SIMULATION RESULTS

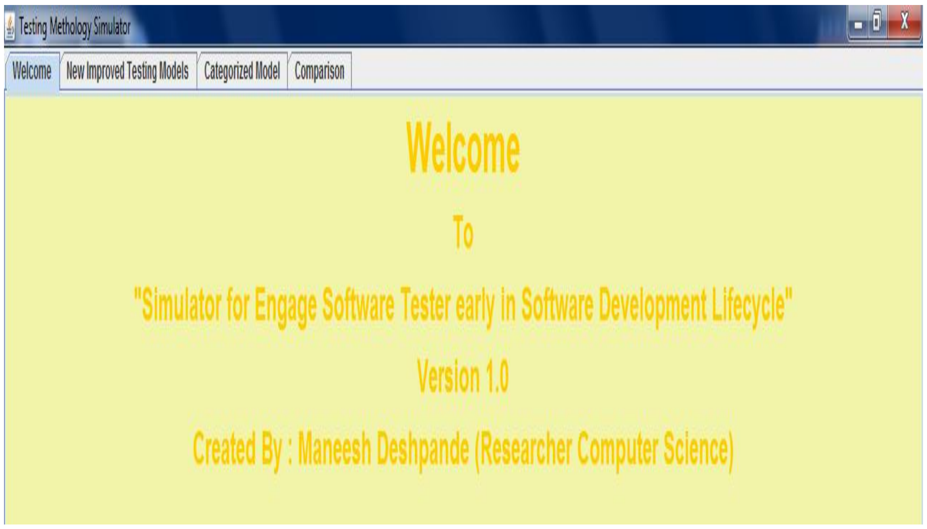
Forsuccessful creation of simulator OrangeHRM Open Source HR Management software used. Our research has focused on OrangeHRM – My Info Module. The reason of choosing the above software is because it's easily available on internet, most of IT companies used the same software and we don't have to pay (Open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

- ✓ Orange HRM 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".



Welcome

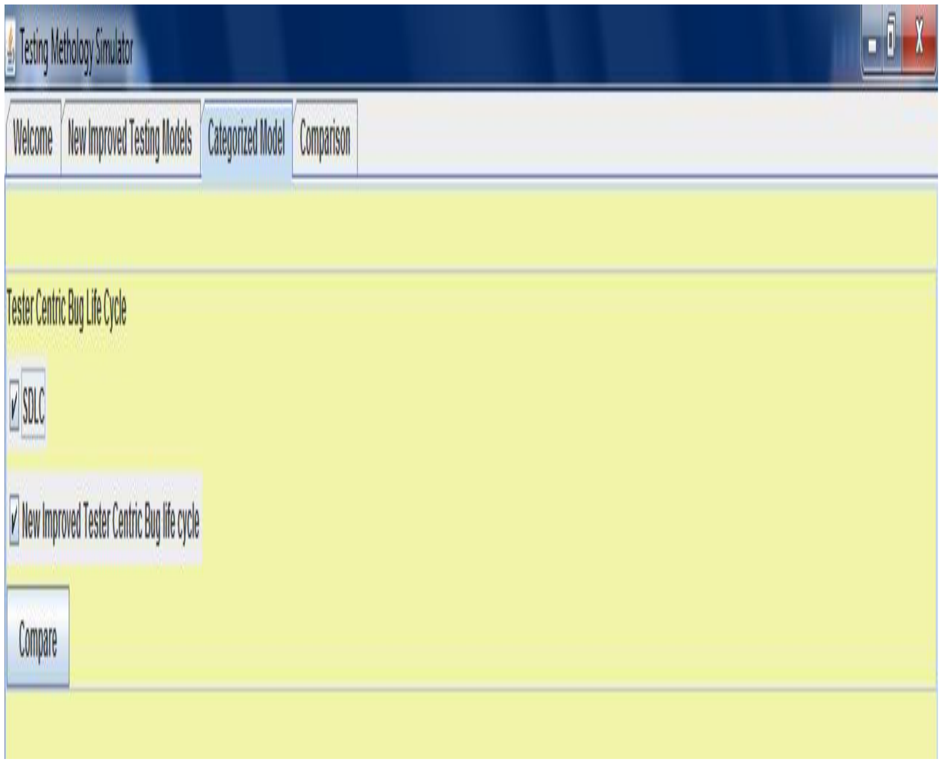
To

"Simulator for Engage Software Tester early in Software Development Lifecycle"

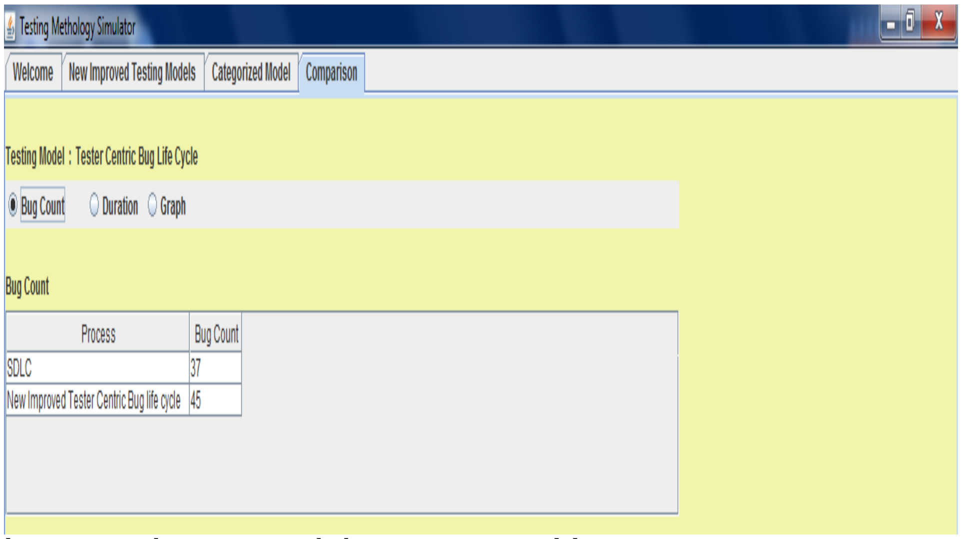
Version 1.0

Created By : Maneesh Deshpande (Researcher Computer Science)

Comparison between New Improved Tester Centric bug Life Cycle and SDLC: Now here comparison took place between new improved tester centric bug life cycle model with normal SDLC. Following snapshot shows the same below.


Testing Methology Simulator
Welcome
New Improved Testing Models
Categorized Model
Comparison
Tester Centric Bug Life Cycle
✓ SDLC
✓ New Improved Tester Centric Bug life cycle
Compare

Bug Count: Following bug count snapshot appears with the two concern models.



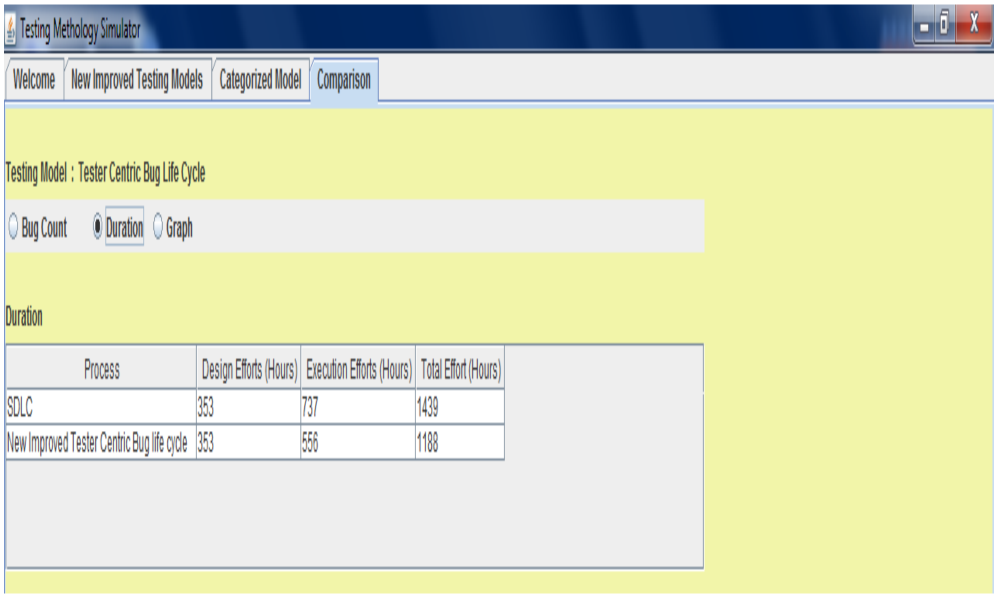
Testing Model : Tester Centric Bug Life Cycle

[✓ Bug Count] Duration Graph

Bug Count

Process	Bug Count
SDLC	37
New Improved Tester Centric Bug life cycle	45

Duration: Following duration snapshot appears with the two concern models.



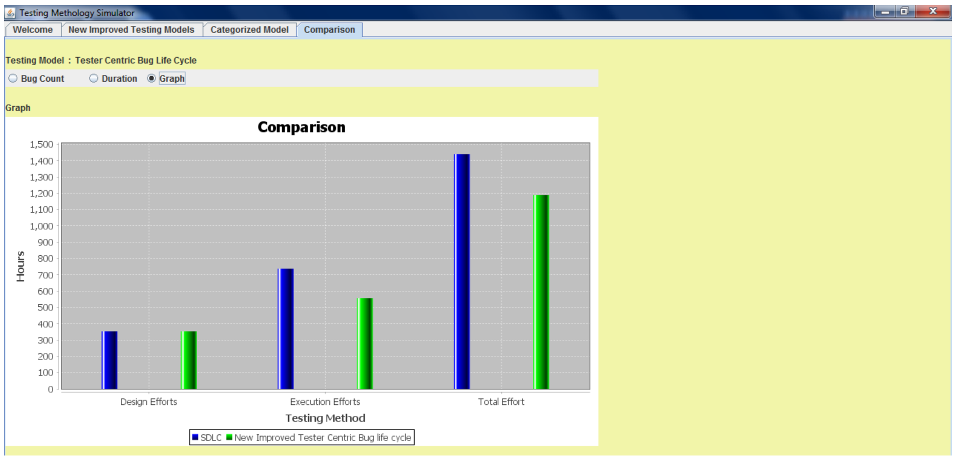
Testing Model : Tester Centric Bug Life Cycle

Bug Count [✓ Duration] Graph

Duration

Process	Design Efforts (Hours)	Execution Efforts (Hours)	Total Effort (Hours)
SDLC	353	737	1439
New Improved Tester Centric Bug life cycle	353	556	1188

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 two concern models.



Testing Model : Tester Centric Bug Life Cycle

Bug Count Duration [✓ Graph]

Graph

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.	New Improved tester centric bug Life Cycle Model	1188	45	353+556+279	*3.78%*

B)Comparison based on Bug Count considering Stated Model: Following table shows total bug count during all the phases of software development. Starting with Normal Software development and comparison with newly created model.

Table 2: Result of simulation based on bug count.

Normal SDLC Bug Count	New Stated Models Bug Count
37	45

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.

Table 3: Comparison based on percentage of Bug Count

Normal SDLC Bug Finding Ratio	New Models Bug Finding Ratio
2.57%	3.78%

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 Company. Since Customer one time wait for software product but can't allow the defected product. While implementing the above stated models certainly software companies required a less time as compared to normal software development as mentioned in following table.

Sr. N	Models Type	Required Time In Hrs.
1.	Normal **Software Development** Life Cycle	**1439**
2.	New Improved tester centric **bug Life Cycle** Model	**1188**

While considering the above table normal SDLC requires time 1439 (in hours) to complete the software development process, but if stated model (Improved bug life cycle) implemented then it requires much less time in the form of 1188(in hours). So time management is reduced in this case. Secondly bug counting is also essential, and tried to improve the same as normal SDLC as shown in following table.

Sr. N	**Models Type**	**Required Time In Hrs.**	**Bug Count**	**Findings**
1.	Normal **Software Development** Life Cycle	**1439**	**37**	*0.025*
2.	New Improved tester centric **bug Life Cycle** Model	**1188**	**45**	*0.037*

So normal SDLC bug counting is 0.025, and newly stated model much more improved as well in bug counting also as 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. In Normal SDLC the same above ratio is 2.57, but stated models ratio is much superior.

Normal SDLC Bug Finding Ratio	**New Models Bug Finding Ratio**
2.57%	3.78%

Finally we can say that the Improved bug life cycle is far more better than Traditional bug life cycle in terms of duration, bug counting, and percentage of bug finding ratio.

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 , Dr. Ujjwallanjewar Dr. Pradeep K. Butey for giving proper guideline 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. IoanaRus , 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 Author: Jeffrey M. Voas Keith W. Miller
[5]. 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)
[6]. Testing and Software Technical Risk Assessments Author: Brad Neal, SimVentions
[7]. Development of Decision Models for Best Use of Software Testing Resources Author: Charles J. Campbell, Judith C. Simon, Ronald B. Wilkes
[8]. Hung Q. Nguyen, Michael Hackett Brent K. Whitlock “Happy About® Global Software Test Automation” Book Excerpt A Discussion of Software Testing for Executives