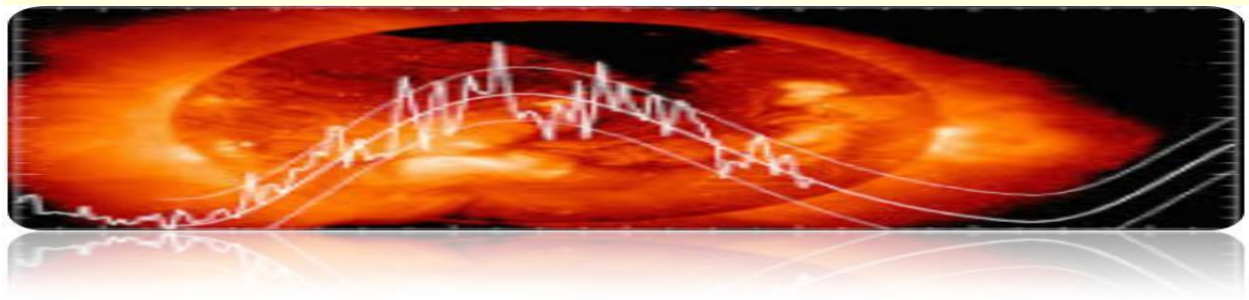
PROGNOSTIC DEFECT MODEL FOR FACTORY MODE AGILE TESTING – AN APPROACH..!!!

Gopalakrishnan Visvanathan*

Neeraj Dhote*



* **Business Value Enhancement, Tech Mahindra**

CONFIDENTIALITY

The information contained in this document is confidential and proprietary information of Tech Mahindra and its affiliates. Any use of this document or parts thereof for any purpose without the express prior written permission of Tech Mahindra is prohibited.

PROPRIETARY NOTICE

This document contains confidential information of *Tech Mahindra,* which is provided for the sole purpose of permitting the recipient to evaluate the contents submitted herewith. In consideration of receipt of this document, the recipient agrees to maintain such information in confidence and to not reproduce or otherwise disclose this information to any person outside the group directly responsible for evaluation of its contents, except that there is no obligation to maintain the confidentiality of any information which was known to the recipient prior to receipt of such information from Tech Mahindra, or becomes publicly known through no fault of recipient, from Tech Mahindra, or is received without obligation of confidentiality from a third party owing no obligation of confidentiality to Tech Mahindra and above specified product vendors.

SECURITY WARNING

The information contained herein is proprietary to Tech Mahindra and may not be used, reproduced or disclosed to others except as specifically permitted in writing by Tech Mahindra. The recipient of this document, by its retention and use, agrees to protect the same and the information contained therein from loss or theft.

© TECH MAHINDRA 2013

This document includes statements which may constitute forward-looking statements made pursuant to the safe harbor provisions of the Private Securities Litigation Reform Act of 1995. Any and all content contained within this document is the property of Tech Mahindra. No portion of any content contained herein may be copied, circulated, quoted or reproduced outside of the Organization without the express permission of Tech Mahindra for any purpose whatsoever.

ABSTRACT:

"One cannot manage change. One can only be ahead of it" – (Peter F. Drucker, 2009)

In this ever changing world we need to stay ahead of ourselves for better today and tomorrow for our partner and customers. Quality is the key for creating values to the customer.

Agile methodology is synonymous to reduced time to market, output based evaluations and stable product, it bring a flexible way to introduce new software products. Testing is an important facet software development and a meticulously approach to testing is essential to success of agile. Testing has to catch up with agile in a transformed approach. Especially when requirements are provided as small user stories needs to develop, test and deploy with short span of time.

In multi vendor & multi geographic location, its a challenge to balance the effort between the project management (per se test management) and core engineering (per se testing) activities and to deliver quality software.

Testimonial of software product/IT services are appraised by number of defects encountered by customer. This paper will bring to you an approach and view(s) from practitioner to analyze defect detection rate and various factors that influence it.

Methodical analysis help you to arrive at the total number of test cases to be written, baseline of defect detection, estimating of testing effort for all the testing phases and business, cost value of defects. Predictive defect model will help test manager and scrum masters to estimate, plan for risk mitigation and formation of test strategy. This predictive model enables collaborative ecosystem among all stakeholders to plan, priorities and execute projects to a greater degree of success.

KEYWORDS:

Defect Forecast, Predict, Agile Testing, Test Factory, Reliability, Rayleigh Curve, Cost of Defect & Business Value Optimization

CONTENTS

Table of Contents

Table of Figures

BACKGROUND/EXECUTIVE SUMMARY

In broad spectrum, software quality assurance framework revolves around two basic concepts:

1. Identifying Defect
2. Removing Defect (Removal effectiveness)

Obviously, defect removal is one of the top expenses (in terms of review effort, testing efforts) and it considerably impacts the schedule and cost of the project. (Raj Prixit, Shrivastava Shailesh, 2012)

Studies and research have also shown that cost of fixing defects is much higher in later phases of the projects/ program and goes up to 100X more, as compared to if it would have been identified in earlier phase. (Quality Improvement Consultants Inc & World -Class Quality)

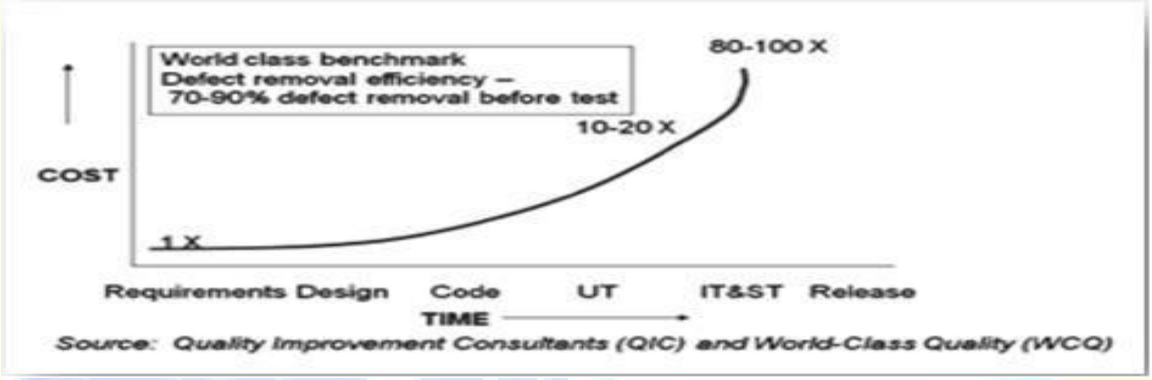


Figure 1: Defect Detection

The above discussion demands that it is important to perform risk analysis in the earlier stage of software life cycle; so that the risks can be foreseen and be prevented or at least reduce the impact of risks if they occurred.

Especially when the projects are developed & tested in agile methodology; most of the time requirements are vague and few user stories are ambiguity in nature. Project team used to form as scrum team for fast communication, giving more emphasis on parallel testing, automation and release. Hence it is always challenge to predict the defect upfront; also estimation may vary from release to release and depends upon various factors.

In such scenarios an estimation with detailed inputs for all the activities and phases plays an important role. Reliability, consistency and diverse option to choose from is the of prime importance. This paper aims to define the "Real reliability" of software product as a "Function **of** time" phases of software testing life cycle (STLC) in factory mode of agile testing. (Raj Prixit, Shrivastava Shailesh, 2012)

KEY TRIGGERS/KEY TRENDS

LIMITATIONS OF EXISTING MODEL

In spite of multiple models and research paper, we do not have software reliability growth models, that caters the needs of all the parameters/ factors and attribute of software development lifecycle (SDLC). There is no good quantitative, analytical methods have been developed to represent software reliability without excessive limitations or assumptions. (Raj Prixit, Shrivastava Shailesh, 2012)

Real Reliability is still not a practiced concept/approach in software, as compared to hardware or system reliability. Whatever software reliability model available today is not considering assumption and limitation of software development while modelling the reliability of the system.

1. Most of the models evaluating software reliability concentrate on defects found during testing for software developed internally and one-off systems, broadly ignoring life cycle defects for multi-release commercial software systems. UAT & production defects are the main reason behind failures in software, by negotiating the total number of defects in each of the SDLC phase and the number of defects that seeps undetected to the next software development lifecycle phase is important in managing software reliability.
2. Software reliability varies with how and in what environment the software is used, which is not factored in any of software reliability studies or research. Unlike defects in a hardware product, software defects are one of its kinds. Specifications and various parameters/ factors can vary from one version to another of the same software or similar software. (Raj Prixit, Shrivastava Shailesh, 2012)
3. There might be differences in the complexity of system and the extent of changes introduced by a release can cause variations in production defect or incident occurrence and its characteristics.

THE SOLUTION/POINT OF VIEW

Objective or intended solution of this white paper is to use real reliability analysis in operational risk modelling for multi-release software system developed and tested in agile methodology. The point of study would be on proposing “Real” reliability, which as an alternative of focusing on UAT & production defects, focuses on detecting the defect via (Agile Testing) as early as they once get introduced. Following provides a framework or mechanism for improving the reliability of application. Key goals are as below.

1. Predicting reliability for all real life assumption and limitations via a new modelling approach by using simulation
2. To streamline and optimized the testing activities is essence to this study. Testing strategy build around will help to enable greater efficiency and effectiveness.
3. Defect forecasting and distribution via Agile Testing Life Cycle Phases.
4. An early indicator of deviation of trend in writing test cases and defect detection.
5. It present an opportunity managing the risks that might arise due to defects in UAT or Production.
6. Optimizing business value of a system and cost by identifying the critical or show stopper defects in earlier phases (which might occur in UAT or Production); by referring forecasting model

COMPONENTS OF THE SOLUTION/POV

DETAILED METHODOLOGY

The software development life cycle (SDLC) process is a continuous process where functionality of product or system is analysed, designed and is then compiled in some programming language, which we refer to as source code. Defects are introduced as the source code is written by the developers. Given the flow of activities it would be appropriate to simulate the reliability of system by modelling in terms of defect detection. (Roger S Pressman, R. S. Pressman, 2010)

This model is based on Rayleigh reliability model and it uses defect distribution. Monte Carlo simulation is been used on multiple factors/ parameters to forecast number of defects its distribution on multiple scenario. (Stephen H. Kan, 2002) (Shunkun Yang, Minyan Lu, Lin Ge, 2013)

- "Real" Software Reliability

Software reliability is defined as '*probability of failure-free software operation for a specified period of time in a specified environment*' by American National Standards Institute - ANSI

Though software reliability is probabilistic in nature, software reliability is not a direct related to time. Hardware is usually associated with the classical reliability theory, which talks about decay in principal over period of time with probability of partial or complete failure. In

software development and testing reliability of system will be on incremental slope as source code will mature with more testing, re-development and retesting.

Rayleigh equation denotes the defects detected in various phases during various life cycle corresponds to a numerical distribution. Rayleigh reliability model widely acknowledged as next to real life prediction model. (Stephen H. Kan, 2002)

- Rayleigh Model

Discovered by the English physicist Lord Rayleigh, in his research corresponding to scattering of acoustic and electromagnetic waves. The same theory applies to software industry. From the time that we start to define, design, write, integrate and test source code, we have the capability of introducing defects in a software product. The Rayleigh function used effectively to denote defect discovery over a period of time. (Stephen H. Kan, 2002)

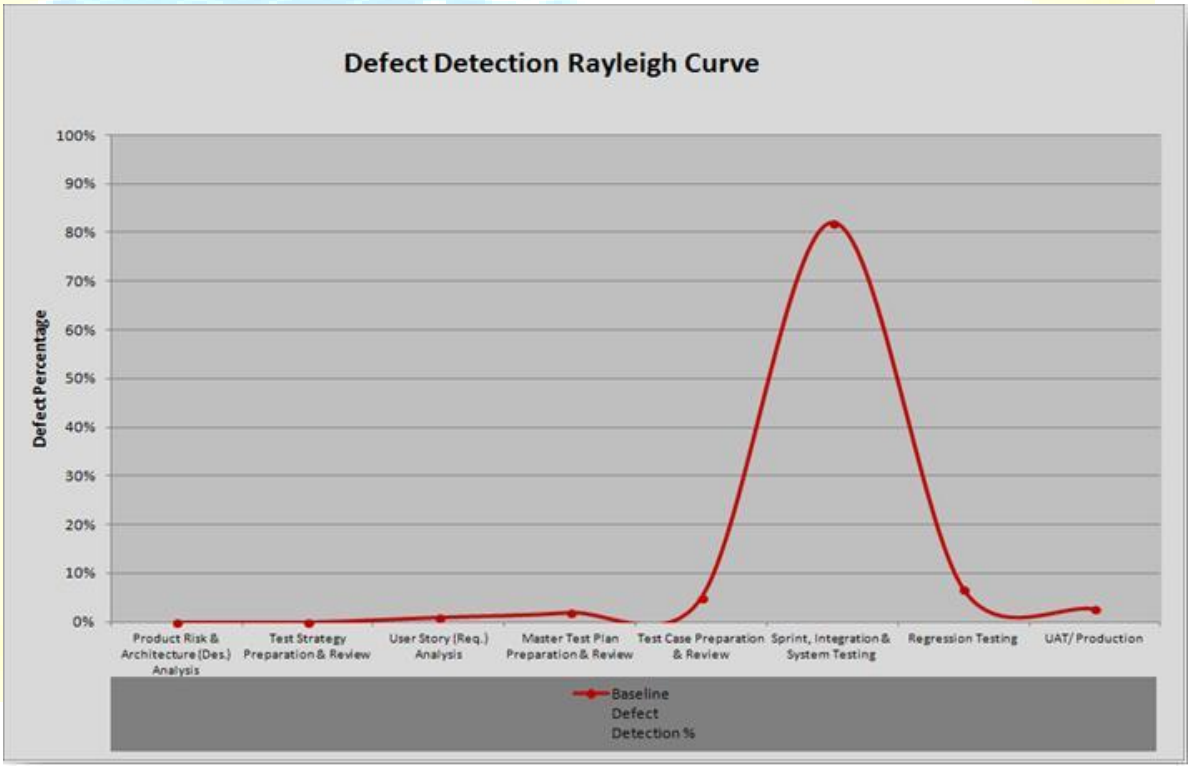


Figure 2: Rayleigh Curve

For forecasting reliability of a software the Rayleigh model is one the most trusted model . Defect density/detection can be predicted at different phases, with various parameters/ factors (total number of defects with respect to time) for the curve are decided. (Stephen H. Kan, 2002)

Y = C*fA*fS*fM*fTE*fAT*fOn*fOf

Y: Forecasted Defect

C: Mean Defect Density (cumulative & calibrate over period of time) fA*fS*fM*fTE*fAT*fOn*fOf: Peak of the curve with various defect determining factors; which detailed in later part of this paper

This presents us with various factors/ levers that we can used on respective phases to monitor and control defect detection.

The given graph Figure 2 confirms the distribution of defects over STLC Phases of Agile Testing as equitant to Standard Rayleigh curve.

- Data Collection and Analysis for Defect Forecasting

A consolidated of 150+ project defect data collection for various life cycle phases for testing of different applications and systems of European major customer.

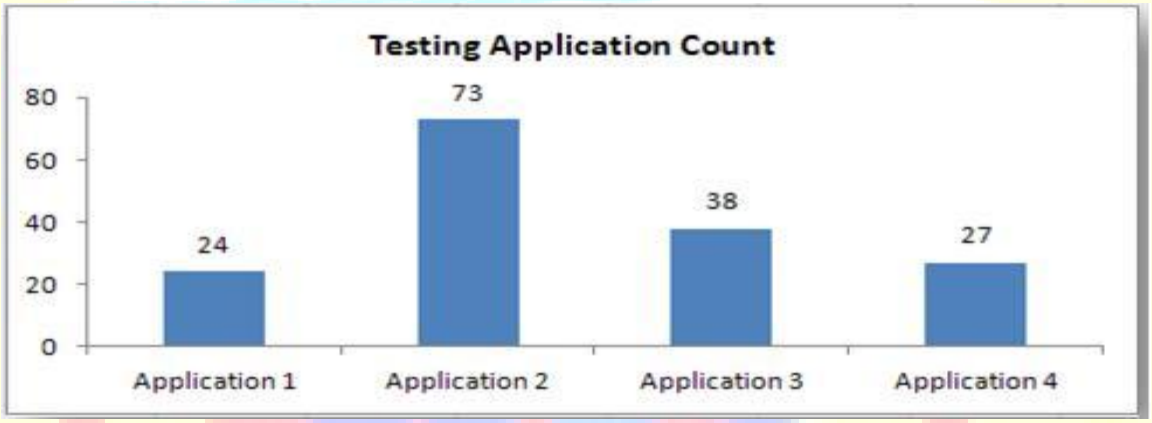


Figure 3: Testing Application Count

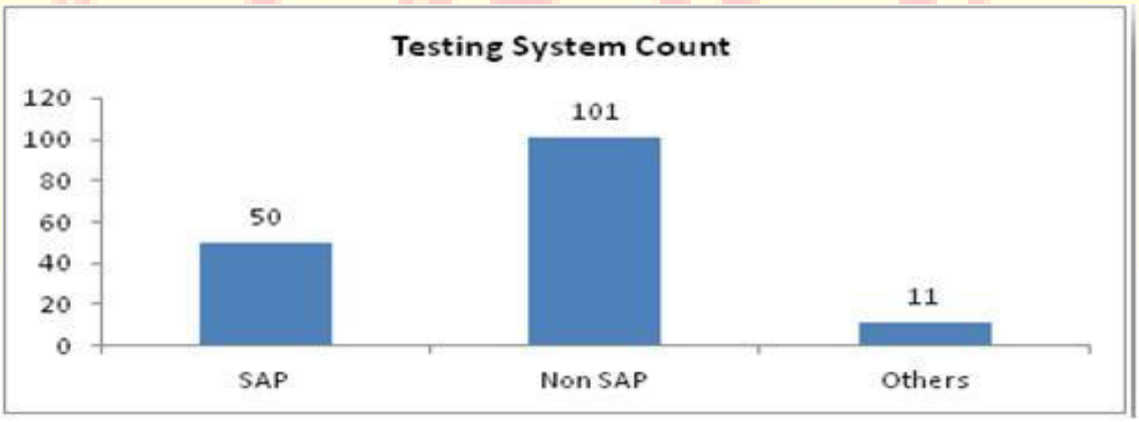


Figure 4: Testing System Count

Define time scale for Agile Software Testing Life Cycle (STLC) activities to help in simulating and modelling. The following time scale is used in the model for carrying out non-linear regression and calculating "Real reliability". (Stephen H. Kan, 2002)

Table 1: Time Scale Modelling

Stage Time Scale	Agile Testing Lifecycle Applicable Phase
1	Product Risk & Architecture (Des.) Analysis
2	Test Strategy Preparation & Review
3	User Story (Req.) Analysis
4	Master Test Plan Preparation & Review
5	Test Case Preparation & Review
6	Sprint, Integration & System Testing
7	Regression Testing
8	UAT/ Production

- **Preparing Baseline**

For each identified category of application and system, i.e., consumer lifestyle, corporate functions, healthcare, lighting, SAP and Non SAP. We have run the descriptive analysis on defect detection rate to calculate mean, standard deviation and 95% Confidence Limit limits. This becomes the baseline for calculating defect detection rate for given input application, system, test effort, average team experience and onsite-offshore ratio.

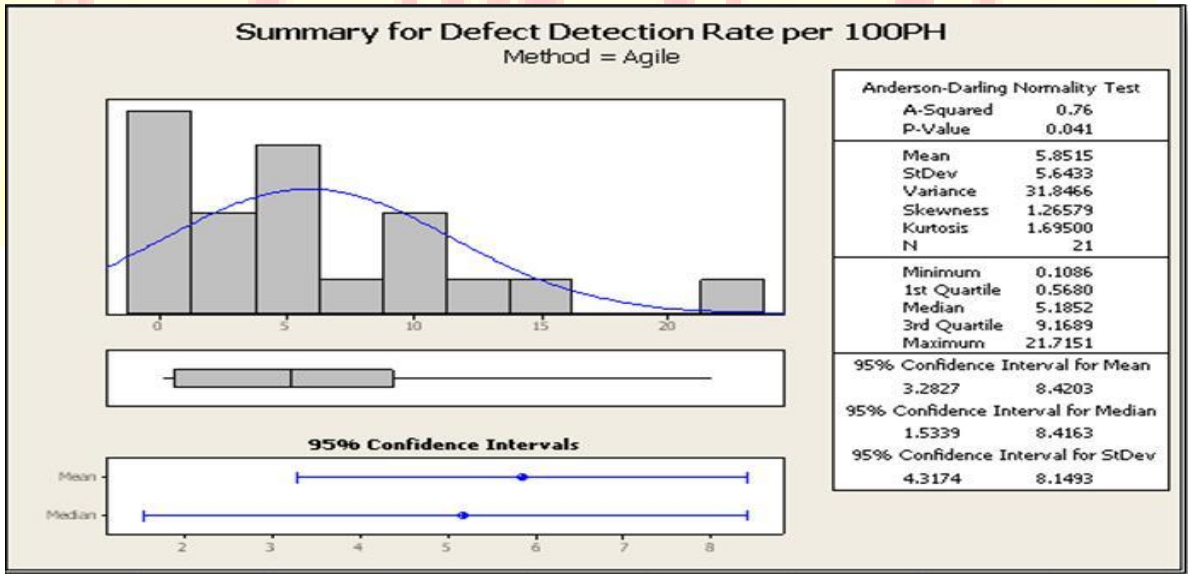


Figure 5: Statistical Analysis for Baseline Preparation

- **Baseline:**

Table 2: Summary of Statistical Analysis

Defect Detection Rate for 100PHrs

S.No	Category	Application	N	Min	P25	Median	P75	Max	Mean	Std Dev	95% Data Range of Median (LSL,USL)	
1	C1	Application 1	24	0.151	1.807	5.188	17.406	51.515	10.988	13.862	2.768	10.065
2	C2	Application 2	73	0.0801	1.5668	5	9.0083	98.5294	8.5852	13.9397	2.4891	6.0192
3	C3	Application 3	38	0.1389	0.758	2.7314	5.8513	65.7534	6.0827	11.2844	1.303	4.8225
4	C4	Application 4	27	0.1086	0.3906	1.6312	8.1871	21.7151	4.4479	5.3765	0.5551	6.028

Defect Detection Rate for 100PHrs

S.No	Category	System	N	Min	P25	Median	P75	Max	Mean	Std Dev	95% Data Range of Median (LSL,USL)	
1	C1	NSP	101	0.115	2.185	5.515	12.425	98.529	10.516	14.516	4.603	8.187
2	C2	OTH	11	0.0801	0.1511	0.655	2.4291	28.5652	3.8632	8.4203	0.1452	2.7849
3	C3	SAP	50	0.0801	0.5056	1.5215	3.7317	20.6897	2.7416	3.5092	0.7936	2.8174

Defect Detection Rate for 100PHrs

S.No	Category	Method	N	Min	P25	Median	P75	Max	Mean	Std Dev	95% Data Range of Median (LSL,USL)	
1	C1	Agile	21	0.1086	0.568	5.1852	9.1689	21.7151	5.8515	5.6433	1.5339	8.4163
2	C2	Project	133	0.0801	0.8475	3.4483	8.5784	98.5294	7.8975	13.1947	2.461	5.1342
3	C3	Release	8	0.4006	1.6616	2.5522	18.6977	29.661	8.5537	11.1765	1.4239	23.0475

Defect Detection Rate for 100PHrs

S.No	Category	Test Effort	N	Min	P25	Median	P75	Max	Mean	Std Dev	95% Data Range of Median (LSL,USL)	
1	A	< 200	48	0.515	2.593	6.712	13.652	98.529	13.413	18.96	5.103	9.818
2	B	201 to 500	46	0.2232	1.259	3.4825	8.1202	32.0557	6.0603	7.0488	2.3548	5.4886
3	C	501 to 1000	27	0.1086	0.3175	2.5373	5.9701	25.5285	4.8227	6.2848	0.8394	5.2651
4	D	1001 to 1500	27	0.0801	0.4006	0.8814	3.6058	29.7747	3.6109	6.5283	0.4006	1.9302
5	E	> 1500	14	0.6988	1.4779	3.0326	7.6271	28.5652	6.5276	8.4314	1.4909	7.2606

Defect Detection Rate for 100PHrs

S.No	Category	Average Team Exp	N	Min	P25	Median	P75	Max	Mean	Std Dev	95% Data Range of Median (LSL,USL)	
1	A	< 4 years	12	0.2353	0.4389	1.4406	7.1906	22.4771	4.7402	7.0736	0.4409	7.1523
2	B	4.1 to 7 years	58	0.0801	0.6488	2.4847	6.9896	21.7151	4.4521	5.1251	1.4968	3.9369
3	C	7.1 to 10 years	37	0.1806	1.216	2.9545	6.5034	65.7534	8.3416	14.8631	2.0011	5.2474
4	D	> 10 years	55	0.123	2.206	6.061	13.772	98.529	11.235	15.608	5.122	9.194

Defect Detection Rate for 100PHrs

S.No	Category	Onsite Ratio	N	Min	P25	Median	P75	Max	Mean	Std Dev	95% Data Range of Median (LSL,USL)	
1	A	< 10%	33	0.0801	0.625	2.5373	7.7317	28.5652	4.7269	5.8008	1.0477	5.7302
2	B	11% to 20%	43	0.0801	0.6988	2.3739	5.5147	65.7534	6.1153	11.2369	1.155	5.0587
3	C	21% to 30%	19	0.4006	2.5316	5.1282	13.2911	23.3333	8.1774	7.5751	3.5575	9.0075
4	D	31% to 40%	25	0.1086	0.8561	4.1096	8.538	51.5152	8.6799	12.7523	1.7006	8.1679
5	E	> 40%	42	0.115	1.519	4.502	13.404	98.529	10.723	17.424	2.15	9.057

Defect Detection Rate for 100PHrs

S.No	Category	Offshore Ratio	N	Min	P25	Median	P75	Max	Mean	Std Dev	95% Data Range of Median (LSL,USL)	
1	A	< 60%	42	0.115	1.519	4.502	13.404	98.529	10.723	17.424	2.15	9.057
2	B	61% to 70%	25	0.1086	0.8561	4.1096	8.538	51.5152	8.6799	12.7523	1.7006	8.1679
3	C	71% to 80%	19	0.4006	2.5316	5.1282	13.2911	23.333	8.1774	7.5751	3.5575	9.0075
4	D	81% to 90%	45	0.0801	0.6918	2.3739	5.4219	65.7534	5.9092	11.0228	1.0961	4.7418
5	E	> 90%	31	0.0801	0.6944	2.7778	7.9208	28.5652	4.9365	5.9232	1.1193	6.2256

❑ **Few Multidimensional Analysis:**

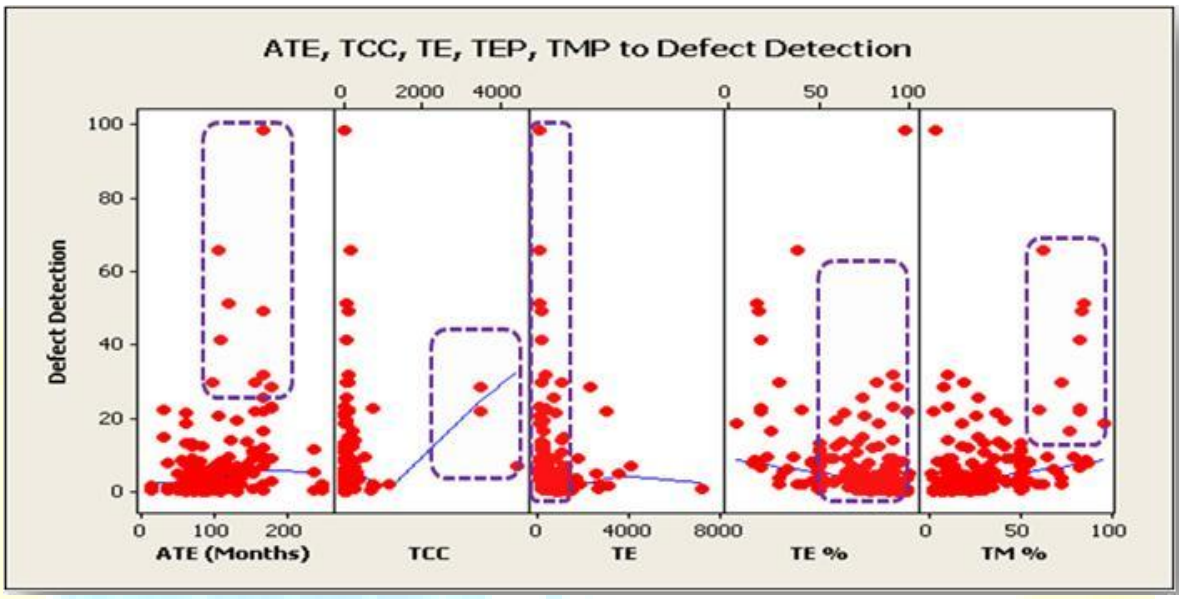


Figure 6: Multidimensional Analysis

- Comparing the Average Team Experience, Test Case Count, Test Effort, Test Effort Percentage and Test Management Percentage;
- If there is a chance to place a few expert with domain knowledge etc in all applications; then it will result in writing more effective test cases covering all test types for defect detection
- Also if we break the projects with less test effort (as like Agile) and then spent hours wisely on test management & testing activities …will result in more defect detection…!!!

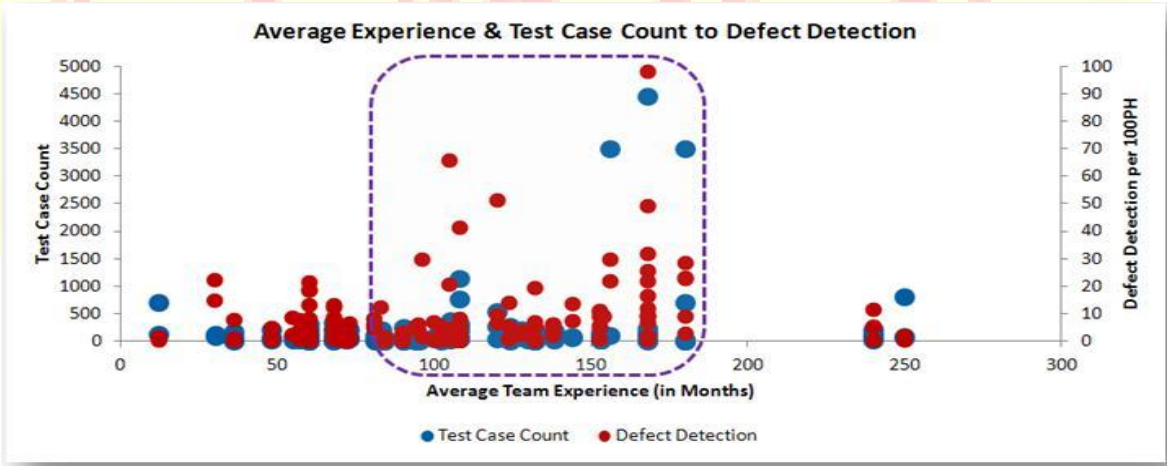


Figure 7: Correlation of Average Team Experience vs. Test Case Count

- If there is a chance to place a few expert with domain knowledge etc in all applications; then it will result in writing more effective test cases covering all test types for defect detection.
- However it affects the EBITDA of Test Factory; hence more caution needed..!!!
  - EBITDA stands for Earnings Before Interest, Taxes, Depreciation and Amortization

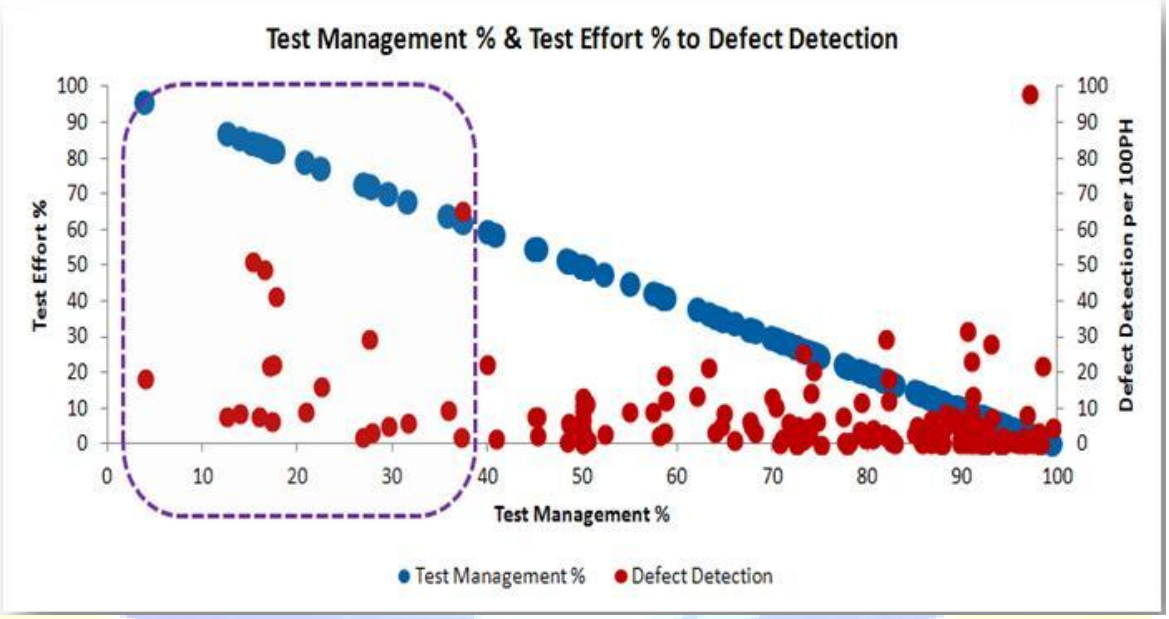


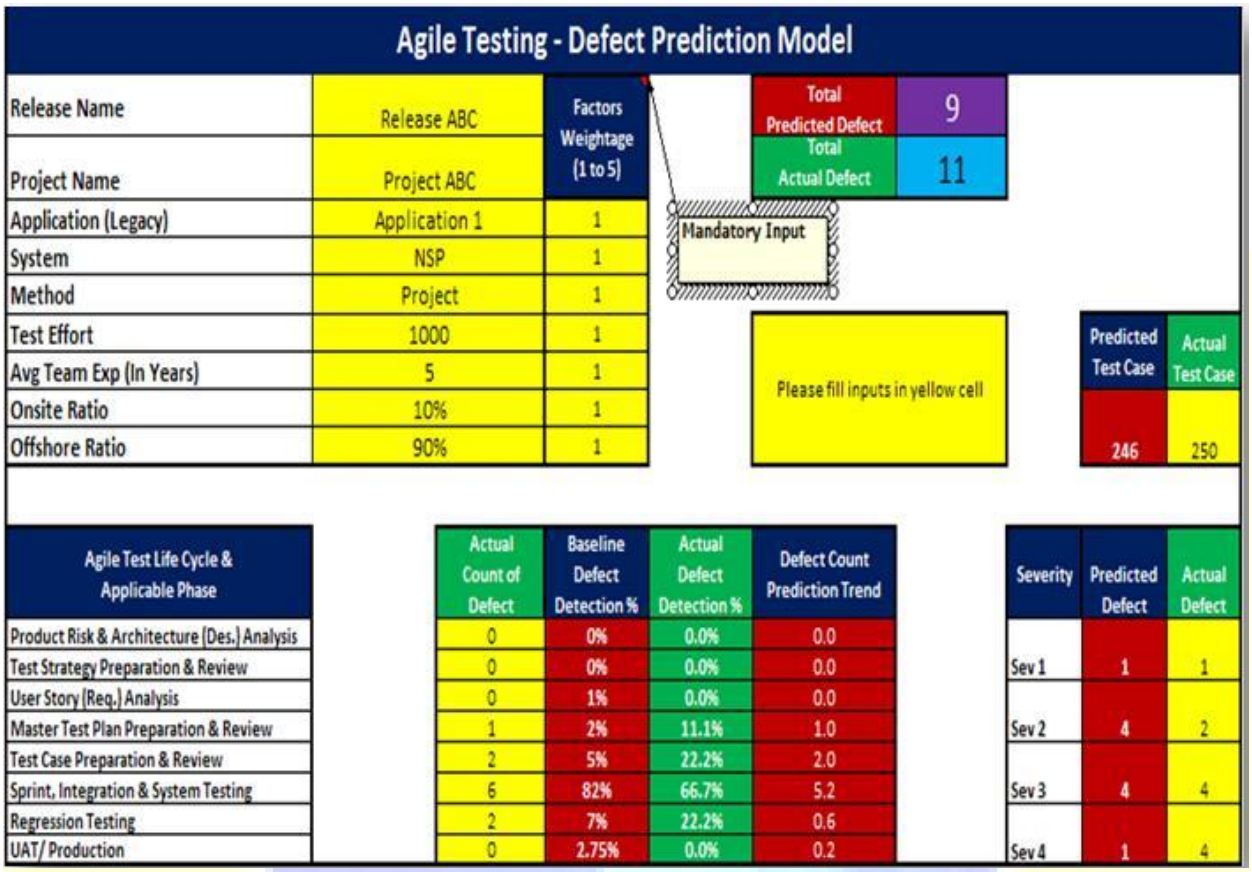
Figure 8: Correlation between Test Management vs. Test Effort

- If there is a chance break the projects with less test effort (as like Agile) and then spent hours wisely on test management & testing activities …will result in more defect detection…!!!

❑ **Prediction using Monte Carlo Simulation:**

Monte Carlo simulation is used to with 100X cycles for simulating values of defect detected in various phases. (Shunkun Yang, Minyan Lu, Lin Ge, 2013); It also showcase the near indication on total number of test cases that testing team would need to prepare and execute. The probable number of defects for the project or release is presented in (i.e., 9).

Table 3: Prediction Tool Input and Output



Agile Testing - Defect Prediction Model

Release Name	Release ABC	Factors Weightage (1 to 5)
Project Name	Project ABC	
Application (Legacy)	Application 1	1
System	NSP	1
Method	Project	1
Test Effort	1000	1
Avg Team Exp (In Years)	5	1
Onsite Ratio	10%	1
Offshore Ratio	90%	1

Total Predicted Defect: 9
Total Actual Defect: 11
Mandatory Input
Please fill inputs in yellow cell

Predicted Test Case	Actual Test Case
246	250

Agile Test Life Cycle & Applicable Phase	Actual Count of Defect	Baseline Defect Detection %	Actual Defect Detection %	Defect Count Prediction Trend
Product Risk & Architecture (Des.) Analysis	0	0%	0.0%	0.0
Test Strategy Preparation & Review	0	0%	0.0%	0.0
User Story (Req.) Analysis	0	1%	0.0%	0.0
Master Test Plan Preparation & Review	1	2%	11.1%	1.0
Test Case Preparation & Review	2	5%	22.2%	2.0
Sprint, Integration & System Testing	6	82%	66.7%	5.2
Regression Testing	2	7%	22.2%	0.6
UAT/ Production	0	2.75%	0.0%	0.2

Severity	Predicted Defect	Actual Defect
Sev 1	1	1
Sev 2	4	2
Sev 3	4	4
Sev 4	1	4

As of now, Seven factors been considered as factors/ levers to run the simulation as

Y = C*fA*fS*fM*fTE*fAT*fOn*fOf

Y: Forecasted Defect

C: Mean Defect Density (cumulative & calibrate over period of time) fA*fS*fM*fTE*fAT*fOn*fOf: Peak of the curve with various defect determining factors

Test Manager has a provision to assign **weightage** for all and or any few of the factors given based on the circumstances and applicability of the program to estimate defect.

Example:

If test manager feels that proficiency of testers (Intermediate) and location (90% offshore) is major factors on the scale of 1 to 5. At least 1 as minimum for remaining factors as a mandatory input.

Then the test manager has provision to modify the factors/ lever as

Y = C*fA*fS*fM*fTE*3fAT*fOn*4fOf

Table 4: Monte Carlo simulation table snapshot

	Application	System	Method	Test Effort	Avg Team Exp	Onsite Ratio	Offshore Ratio	TOTAL
1000	0.78	1.27	0.91	0.51	0.17	0.47	1.60	0.36
Mean	0.54	1.18	0.77	0.51	0.61	1.21	1.20	1.44
Standard error	0.01	0.02	0.01	0.01	0.01	0.02	0.02	0.08
Median	0.55	1.19	0.78	0.52	0.60	1.23	1.19	0.55
Standard deviation	0.24	0.48	0.37	0.27	0.30	0.58	0.55	2.61
Variance	0.06	0.23	0.14	0.07	0.09	0.33	0.31	6.81

- **Output:**

Total estimated numbers of defects are then plotted on a Rayleigh curve against the standard Rayleigh distributions. To do a comparative analysis we need enter actual number of defects for each of the phases, this is represented by a green curve. Ideally green and red curve should be the same . If any significant deviation is observed we need to do a root cause analysis and document the findings as input to calibrate the tool over period of time. Objective is to not seep the defect to UAT or production.

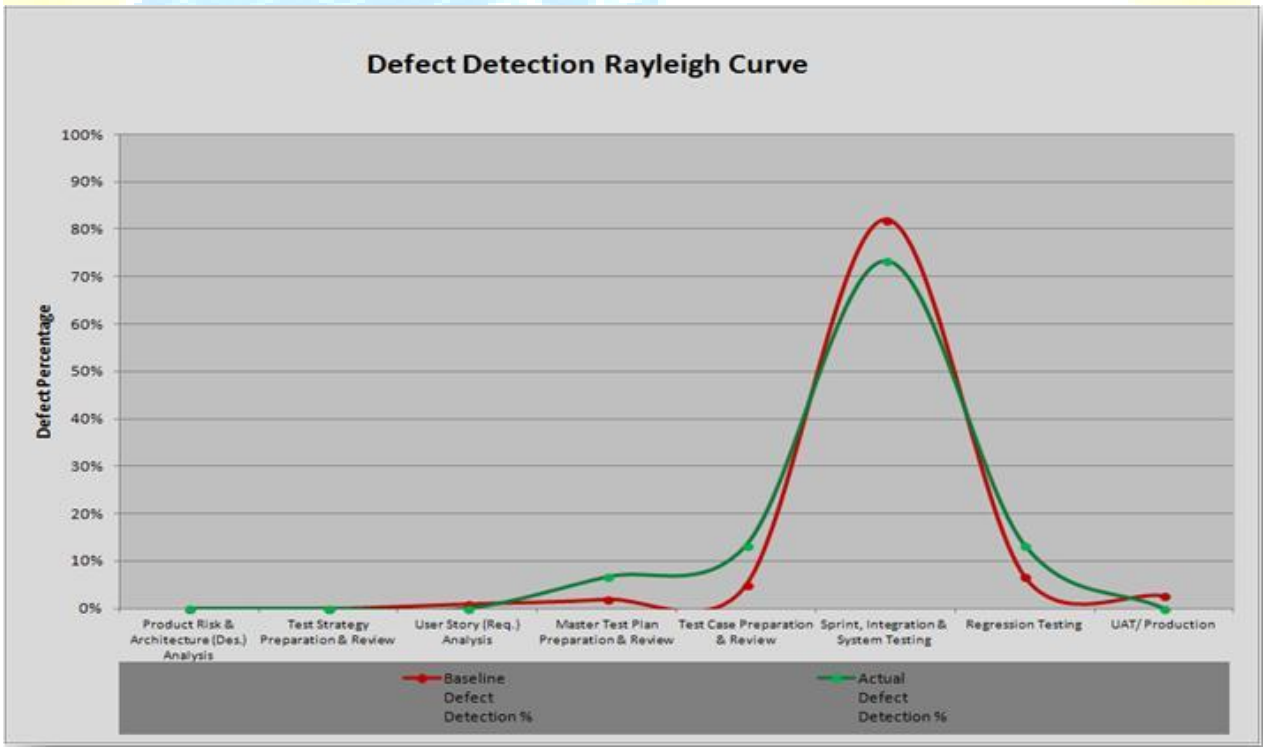


Figure 9: Defect Distribution Rayleigh Curve

Below Radar chart shows probable defect seepage to subsequent phases. We assume that attributes and factors remains as is actual , i.e., number of actual defect defected by testing team would near match with predicted defect.

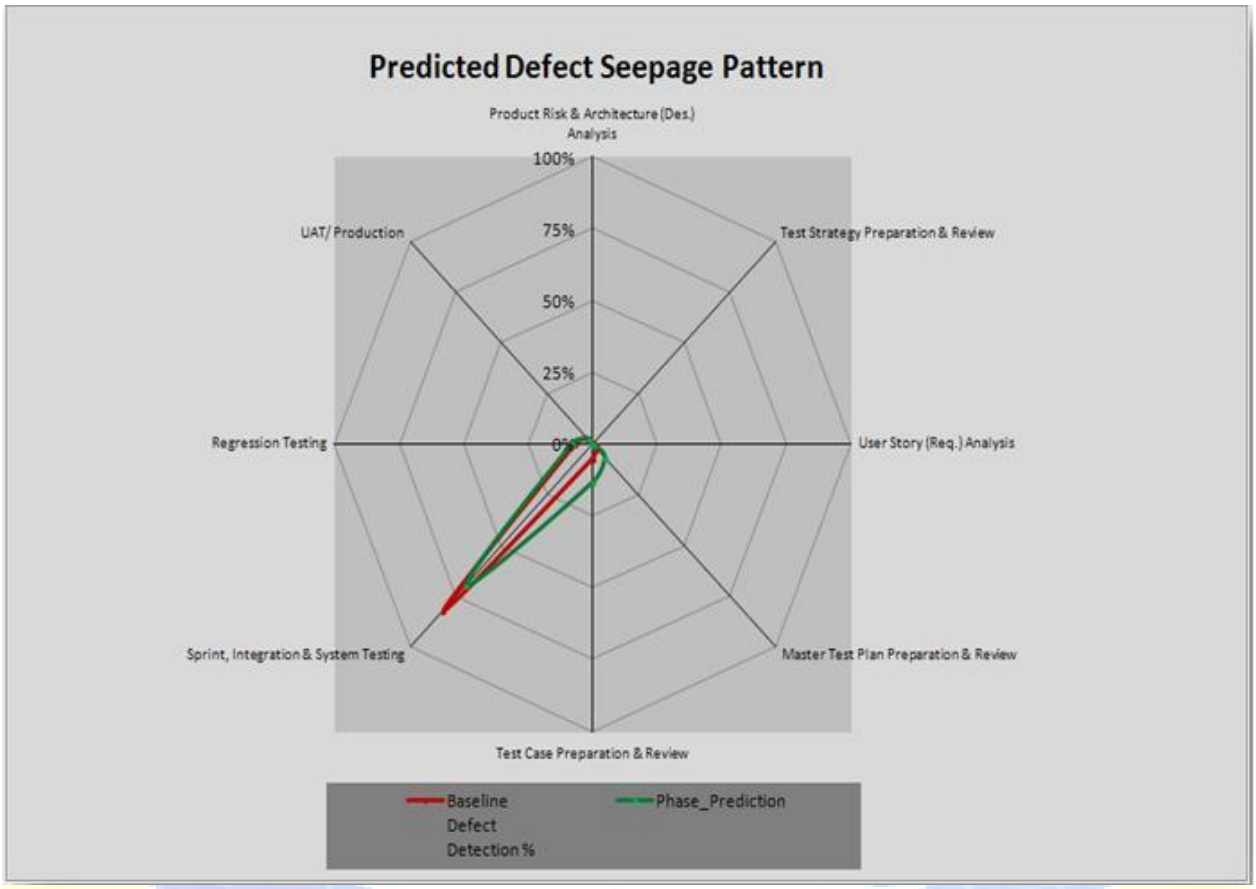


Figure 10: Predicted Defect Seepage Pattern

- **Cost of Defect and Business Value Optimization**

Cost, Business value of defect is mean to showcase the importance of testing and detecting defects in early of Agile Software Testing Lifecycle (STLC) phase. As per software engineering economics , cost to fix a defect in UAT or production is getting higher and hence it is prescribed to have robust testing strategy in agile testing methodology. (Barry W. Boehm, 1981)

It also brings another perspective of looking at over cost of project or release, as this would predict total cost of UAT and production defects that an project end up owning. In most of the cases UAT and production support cost is not efficiently planned.

It will also help project teams to bring down the cost as they can adopt bottom up approach; with the help defect forecasting model at various phases.

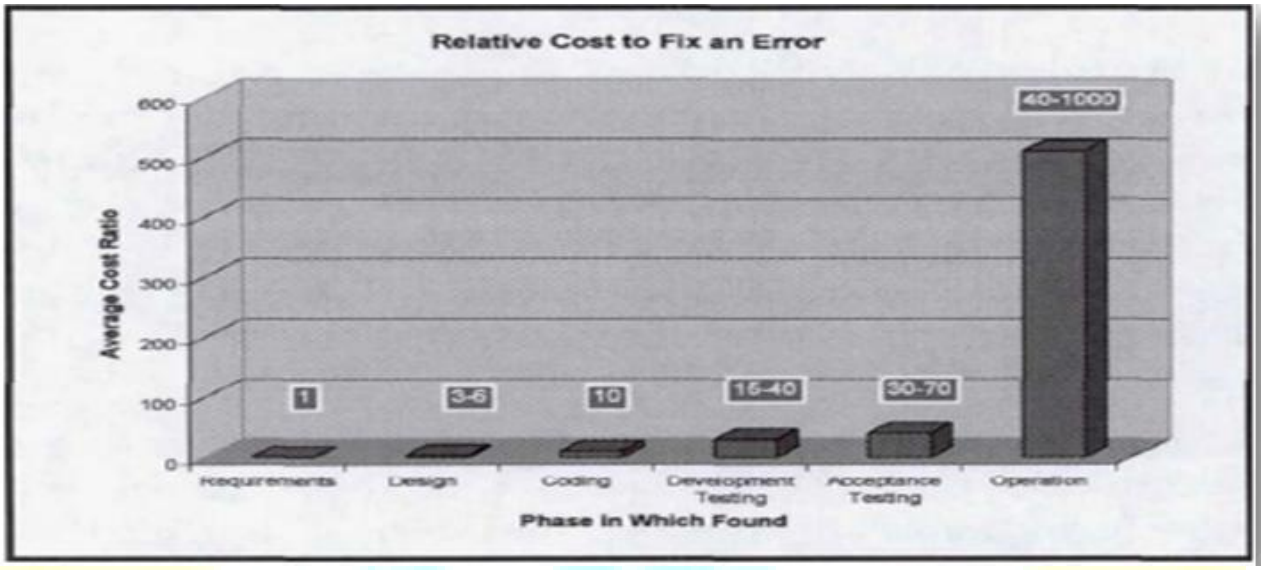


Figure 11: Relative Cost to Fix an Error

Source: Software Engineering Economics by Barry Boehm

❑ **Results:**

Actual values of reliability matches with forecasted project data. Upon plotting predictions and actual values on scatter plot, all our points fall on or nearer to the line; this implies that forecasting is close to actual values. Also it has been concluded through hypothesis test that there is a significant relationship between defect detection and given parameters/ factors to detect defect.

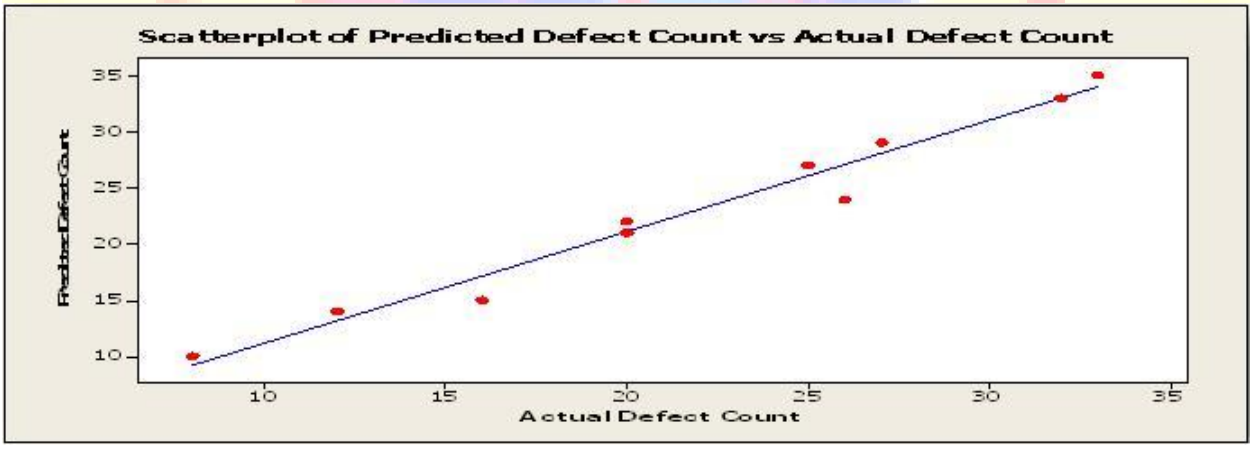


Figure 12: Scatter plot of Predicted vs. Actual Defect

1. Model simulates real life scenarios in agile testing and is scalable, it accounts for various inputs factors and simulates impact of inputs/variables.
2. Allows examining system behaviour under different scenarios.

3. Defect forecast presents an opportunity for reducing the impact of UAT and Production defects
4. Provides near accurate estimates of defects detected across the phases.
5. Provides test cases needs to be written and severity of defects for the release.
6. Provides management a cost, business value optimization with respect to defects across phases.

ACRONYMS

1. UT - Unit Testing
2. LSL - Lower Specification Limit
3. USL - Upper Specification Limit
4. ATE - Average Team Experience
5. TCC - Test Case Count
6. TE - Test Effort
7. TEP - Test Effort Percentage
8. TMP - Test Management Percentage
9. NSP – Non _SAP
10. OTH – Others
11. SAP – SAP
12. IT – Integration Testing
13. ST – System Testing
14. Req. – Requirement
15. Des. – Design
16. Reg. – Regression
17. UAT – User Acceptance Testing
18. Prodn. – Production
19. UCL – Upper Control Limit
20. LCL – Lower Control Limit
21. PH – Person Hours
22. 100PH – 100 Person Hours

APPENDIX

- **Presentation of Statistics**
  - This section briefly describes the tables in the report
  - Through this report each table indicates the number of projects represented (N). This number is important, because care must be taken not to draw unwarranted conclusions from small sample of projects. The number of project varies
  - ***When your are using the tables and charts, always check the number of projects represented. Don't jump to conclusions based on a small number of projects***
- **Explanation of tables**
  - In this report tables are frequently used to summaries some key statistics about the data being analyzed. The following header is common to the tables presented;

Category	N	Minimum	P25	Median	P75	Maximum	Mean	Std. Dev	95% Data Range of Median

  - Category is the type for the projects grouped together
  - N is the number of projects or data instances in the sample
  - Minimum is the minimum value found in the sample
  - P10 is the 10th percentile and is that value which is greater than the values of ten percent of the members of the sample
  - P25 (often also often written as Q1) is the 25th percentile or first quartile. It is that value which is greater than the values of twenty-five percent of the members of the sample or sub-sample
  - Median (sometimes written as P50) is the middle value, half the values in the data sample or sub-sample are below this value, while the other half have values which are greater.
  - P75 (often also written as Q3) is the 75th percentile or third quartile. It is that value which is greater than the values of seventy-five percent of the members of the sample or sub-sample
  - P90 is the 90th percentile and is that value which is greater than the values of ninety percent of the members of the sample or sub-sample

- Maximum is the maximum value found in the sample
- Mean is the arithmetic mean or average
- Std Dev is the standard deviation
- 95% Data Range of Median is that 95% of projects within particular data range in that given sample

- **Usage of Statistics**
  - In most cases we have focused upon the median rather than the mean. The median is the more useful measure when the data contain outliers or when they are strongly skewed
  - ***Mean is the average of all the values***
  - ***The Median is middle value of all the values***
  - Using the mean or average can be misleading when the data is skewed. One huge number can distort the mean, so that it is no longer a fair representation of "average". This is common in software engineering data sets, so the median is usually preferred

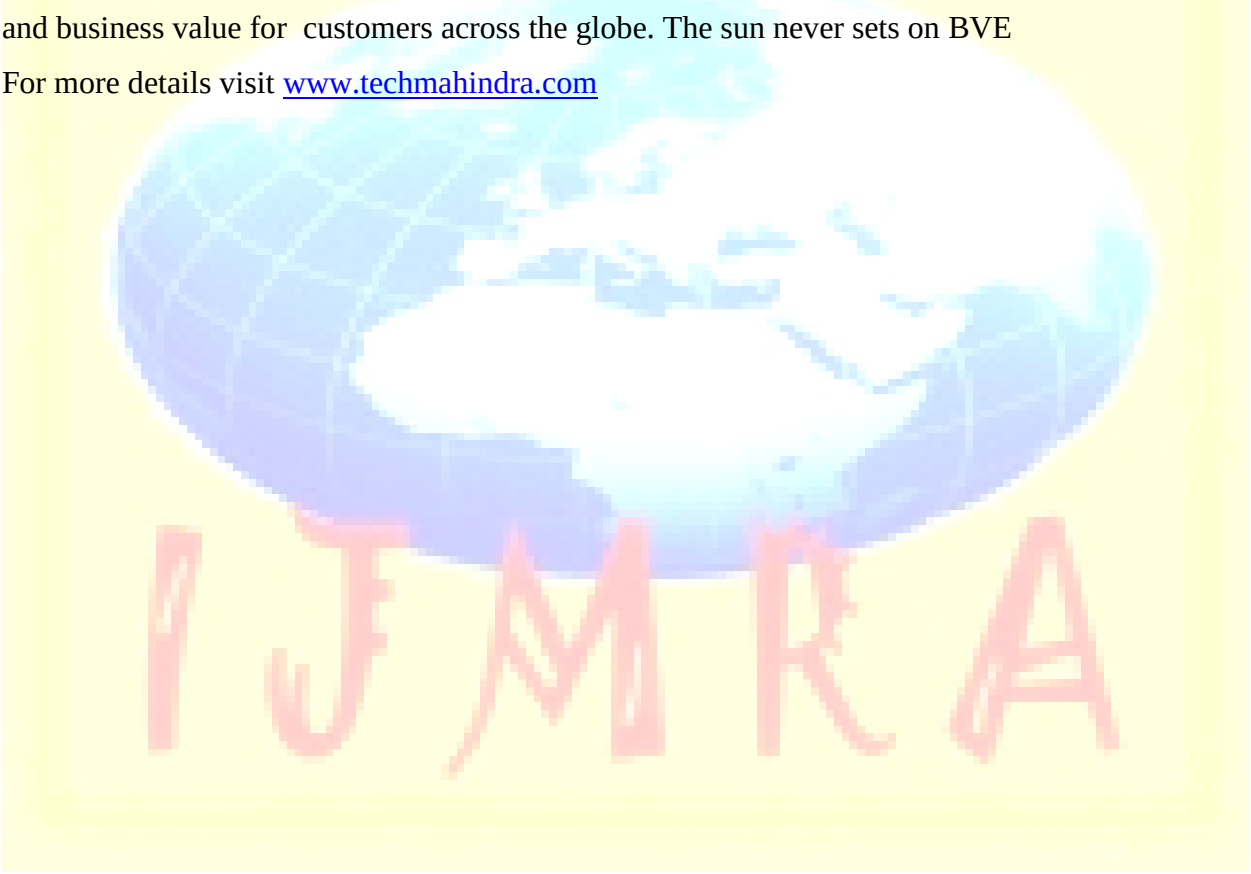
BIBLIOGRAPHY

1. 1981. Barry W. Boehm, Software Engineering Economics. Englewood Cliffs, NJ : Prentice-Hall Publication, ISBN 0-13-822122-7.
2. 2002, Stephen H. Kan, Metrics and Models in Software Quality Engineering, Addison Wesley Publication, ISBN: 0-201-72915-6.
3. 2009, Peter F. Drucker, Management Challenges for the 21st Century, HarperCollins Publication, ISBN: 0887309992.
4. 2010, Roger S Pressman, R. S. Pressman, Software Engineering: A Practitioner's Approach, McGraw-Hill Publication, ISBN: 0073375977.
5. 2012, Raj Prixit, Shrivastava Shailesh, Software-Inlife Reliability Modeling using Simulation, International Journal of Applied Research on Information Technology and Computing Publication, Print ISSN : 0975-8070, Online ISSN : 0975-8089.
6. 2013, Shunkun Yang, Minyan Lu, Lin Ge, Software Security and Reliability (SERE) & IEEE 7th International Conference Publication, Print ISBN:978-1-4799-0406-8.

ABOUT BVE

Business Value Enhancement (BVE) Consulting group of Tech Mahindra helps customers build high performance business operations, make smart investments in Information and Communications Technology (ICT) assets and enhance business value of ICT investments. A team of 350+ consultants help clients achieve breakthrough improvement in business processes by making smart investments that are in line with the strategic objectives of the enterprise. This consulting practice zeroes in on business operations optimization and improving the overall IT function. With an experience spectrum of 600+ engagements, over 15 years of experience in business and IT transformation consulting, this team specializes in tangible business outcomes and business value for customers across the globe. The sun never sets on BVE

For more details visit www.techmahindra.com



BIOGRAPHY

Gopalakrishnan Visvanathan

Gopalakrishnan Visvanathan is a seasoned Process Consultant of ICT Advisory & Transformation Consulting, Business Value Enhancement Team at Tech Mahindra with 10 years of experience in Industry. He has done critical assignments to Fortune 100 clients across globe with various model and non model based frameworks on Business Process Improvements. He is a certified Six Sigma Black Belt from ASQ, Scrum Master from Scrum Alliance, Estimation Practitioner from COSMIC & IFPUG, ITIL Foundation from EXIN and Software Quality Analyst professional from QAI. He holds an Engineering degree from Bharathiyar University and Masters in Business Administration from University of Madras.

Acknowledgements: Thankful to Prakashkumar for motivating, inspiring and challenging the thoughts to write and pack to next level. Also pleased to Neeraj Dhote who always stand next on bringing ideas from out of the box. Last but not least, also like to thank Satyajit Acharya, Sujoy Sen, Habeeb Mahaboob and all fellow colleagues for continual support.

Neeraj Dhote

Neeraj is a Process Consultant of ICT Advisory & Transformation Consulting, Business Value Enhancement Team at Tech Mahindra with 7 years of experience in the Software Industry. He executed assignments across frameworks , domains and continents for various client at Tech Mahindra. He is a certified internal quality auditor and trained agile expert. He holds an Engineering degree from State Technological University of Madhya Pradesh and Masters in Business Administration from Christ University.

Acknowledgements: Thanks Prakash for leading, inspiring and challenging us. To all colleagues and friends who helped us achieve this, for being our sounding board and supporting us.