International Journal of Engineering & Technology, 7 (3.4) (2018) 235-242

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper



A Case Study on Defining a "Quality Algorithm" Based on Correlation between 'Existing Quality Model, Different Attributes of Defects & Tests'

Abhishek Anurag[1]*, R. Kamatchi[2]

[1]*Research Scholar, Amity School of Engineering & Technology, Amity University Mumbai, Maharashtra, India*
[2]*Head, CSE, Amity School of Engineering & Technology, Amity University Mumbai, Maharashtra, India*
**Corresponding author E-mail: anuragabhishek@gmail.com*

Abstract

Usage and nature of software systems have changed significantly. Due to this complexity of software systems has also grown exponentially. In these ever-changing requirements and environment in which software system is being used, maintaining quality of software system is very challenging and difficult. If user requirements are not met as expected, it's called defect. To improve quality, it's critical to understand and analyze these defects. In this study root cause analysis technique is used to analyze defects and their attributes, root cause of defects and corrective actions of defects. A quality model is designed based on defects, root cause of defects and tests. A quality algorithm is designed in this study depending on existing quality model, defects, tests and their attributes. This quality algorithm is executed on a software system to validate quality model. The results obtained are analyzed to understand the quality of the software system and how it's different than existing quality model.

Keywords:*Software quality; Defect attributes; Root cause analysis; Software quality model; Software quality algorithm.*

1. Introduction

Quality is one of most important aspect of software. Quality had been defined in multiple ways and can be understood as: "Capability of a software product to conform to requirements" [2]. For most it's also viewed as "Customer Value" or "Defect Level" [1]. Two aspects of quality are very common: one is – "quality as an Objective reality independent of the existence of human" and another is – "what we as human think, feel or sense because of Subjective side of Quality" [3]. Feigenbaum [4] very aptly explains that "quality is not something management team determines, it's not something marketing team determines, it's not something engineering team determines; rather it's determined by customer". He explained that quality is based on customer actual experiences with the software product or services measured against his or her requirements – stated or unstated, conscious or unconscious, objective or subjective. Quality is having multiple meanings and two of them as per Juran [1] which widely used are: 1. Quality consists of those product features which meet customers requirement, and 2. Quality consists of freedom from deficiencies and can be easily understood as "fitness for use". If a product or services are not fit for use or having deficiencies in them then end user experience is considered as poor and so, do quality. These deficiencies or issues are tracked as defect in software.
Software development process is driven by three factors mainly: interval, quality and cost [5]. Defect is something which impacts all these if occurred. To reduce interval of software design, development and delivery, improve quality and hence the cost, it's needed to root cause the defect reported by customers. Every defect reported is having root cause and based on that corrective cations are taken. These corrective actions can help improve quality of the product and if implemented further during different software stages, defects having similar root cause won't appear again [7]. Root cause and corrective actions are associated with different attributes of defects, tests and software stages and it overall impact the quality of software system [6].
This case study focuses on finding a weighted correlation between all known impacted attributes of defects associated with root cause and corrective actions, tests and software stages and find a mathematical model to define quality.

1.1. Overview of Software System

This case study is based on an Android based very complex embedded system software product. For sake of privacy all sensitive information are kept anonymous and so pseudonymizing has been done. Let's say: the organization in which study was made is called "Company X". Product under study is called "Product X" – one of world's most advanced streaming device. The public software release made to customers over the air (OTA) is called "Release X" for "Product X". Total number of engineers assigned on this product was 761 including 70 from test engineering team and remaining from different aspect of software teams viz. marketing, sales, architect, development, etc. Product X took almost 24 months from inception to first market launch. This product was categorized into 10 different modules and 926 overall software components. Total number of files are 588K, total lines of code are 241 million LOC and total statements documented in the code are 38 million LOC. This product had made 6 major and 10 minors over the air (OTA) public releases to customers [6].
As part of this study total 20794 defects of the Product X were taken into consideration. These defects were in the span of 48

Copyright © 2018 Abhishek Anurag, Prof. (Dr.) R. Kamatchi. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

months [7]. These defects were tracked in Company X defect database management system. These were exported into Microsoft Excel 2016 and all analysis were done based on Excel based functionalities, tables, bar & charts. These defects are analyzed based on different attributes. These are attributes which basically represent at what extent customer's experiences are impacted. The most important aspects of these defects are root cause analysis (RCA) which tells what are root causes of defects and what are corrective actions to avoid such issues in future.

As part of this study all tests of test database management system were also analyzed. Total Android based 23519 tests were analyzed from test database. In this study focus was to analyze tests which are applicable for "Product X" and what are different test types [6].

1.2. Scope of this Study

Scope of this study is to analyze all attributes of defects, tests & software stages and find a way how software quality can be defined mathematically. As part of previous studies few attributes were already studied and analyzed [6, 7]. This study is focusing on below attributes of defects and tests:

- Defect regression
- Defect disposition
- Defect module
- Test types

This study is focusing on identifying the weight associated with every attribute of defects, co-relation between them and finding a mathematical model to define software quality.

1.3. Objective of this study

Objective of this study is to analyze tests, test types, defects attributes and how root cause of customer defects are related to rest of defect attributes. Focus is to establish a mathematical correlation between these attributes. Find a way to create a weight against each attribute and design a mathematical model or an algorithm which will tell the quality of software system. Major objectives of this study are:

1. Analyze 23519 tests available in test database management related to Android embedded software system products.
2. Align and analyze tests related to "Product X" under study.
3. Classification of tests based on test types in [6].
4. Analyze other attributes of defects which might impact software quality:
    a. Defect regression
    b. Defect disposition
    c. Defect module
5. Analyze Root cause of defects which are of origin "Tester" and "Developer".
6. Software stages of defects which are of origin "Tester" and "Developer".
7. Create a mathematical weighted correlation between all attributed and create a mathematical quality model or algorithm based on these.

1.4. Methodologies

This study is based on causal research (experiment based) [6]. The dependent variable under study is "quality score" and independent variables which are of interest in this part of study are: defect regression, defect disposition, defect module, test type & software stages. Based on these variables and other variables analyzed as part of study made in [6, 7] define a quality model and compute what is quality score of product under study, Product X and Release X.

1.5. Related Work

This study is extension of mainly two studies:

1. "Study on Software Quality improvement based on Root Cause & Corrective Actions" [7].
2. "A Case Study of existing Quality Model based on Defects & Tests Management of Embedded Software System" [6].

'Study on software quality improvement' is mainly based on customer defects, root cause of defects and what are systematic corrective actions taken. Different related root cause analysis techniques were analyzed, and it was defined that what are different root cause and corrective actions which when taken on different software stages will help improve quality. Here attributes under consideration were: customer defects, root cause of defect, corrective actions of defects and different software stages – and how all these define a software quality model [7].

'A case study on existing quality model' was mainly based on analyzing all customer defects and differentattributes associated with these. In this case study defects of different origins were analyzed. Based on these defects origin total defects classification was done in three major area: "Developer, Tester & Customer". Defect type or severity was defined, total different defects priority was analyzed. In this study only, customer defects priority was analyzed. In this case study test types were defined, and tests were mapped with different major modules.

Here root cause category definition was provided and associated corrective actions also identified. Also, for different root cause of customer defects which are associated with software stages were also studied [7].As part of case study on Company X, Product X and Release X, what is existing mathematical quality model is also designed [6].

2. Data and Analysis

2.1. Defect Regression

These are defects which are mainly associated with regression testing. Regression testing is like running a test suite named "regression test suite" which checks whether a feature which is known, and working is still working or not. If a working feature is now not working is called a regression and defect which is associated with this is called regression defect. So, regression defect is a defect which caused something which was working fine and correctly, are now not working due to certain events and conditions [8].

This kind of defects are very annoying as it impacts user experience badly and are escalated by customers. These are defects which are hard to reproduce and fix as it misses from development and testing phases. Most of time these defects appear because of new feature implementation, design change or side effect of another defects got fixed and proper validation or regression test suite is not executed. There are basically 3 classifications of defects in terms of regression:

1. Regression – these are defects which are based on earlier feature or functionality which was working and somehow now it's not working.
2. Not a regression – these are defects which are seen for the first time. First time feature or functionality are found not working.
3. Undetermined – these are defects which are neither regression nor not a regression. By considering nature of these defects, it can't be told whether functionality was ever working or first time it's not working.

Product X defects are analyzed and put into regression bucket in Table 1. In this table defects are categorized based on defect origin and defect regression. Almost 64% of defects are not a regression, 16% of defects are regression and 20% defects are undetermined.

Table 1: Defect Origin vs. Defect Regression

Defect Origin		Defect Regression				
May2013-April 2017	Teams	Not a Regression	Regression	Undetermined	Total Defect Regression	Total
Developer	Core Engineering Team	782	86	2411	3279	3369
	Dev Automation Team	13	10	40	63	
	Tools Team	2	0	4	6	
	Application Developer Team	7	1	5	13	
	Build & Infra Team	0	4	1	5	
	Production Team	0	0	1	1	
	Research Team	0	0	1	1	
	White Box Testing Team	1	0	0	1	
Testing	Core Testing Team	8950	2311	833	12094	15030
	Test Automation Team	1131	608	39	1778	
	Test Certification Team	235	161	247	643	
	Driver Test Team	178	110	211	499	
	Hadware Test Team	5	2	7	14	
	Customer Customization Test Team	1	0	1	2	
Customer	End User [Non-Dev & Non-Testing]	1452	86	189	1727	2395
	Customer [Outside Company X]	522	9	55	586	
	Marketing Team	10	1	71	82	
					Total	20794

Figure 1 reflects that out of 16% regressions at customer end 0.46 % defects were only regression. Rest were caught in-house either at developer end or by testers. These 0.46% defects are the ones which usually annoy customer very much as for them what was working earlier is somehow not working. These defects are the ones which impact experience and hence the quality more than any other defects.

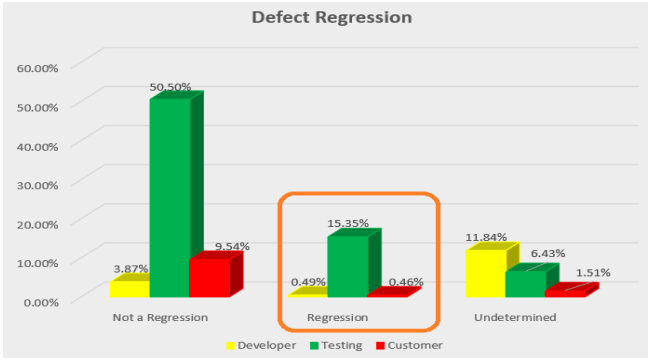


Fig. 1: Defect Regression

In this study it was focused that how these defects regression is associated to software quality and how fixing them after root cause of such defects are going to help improve quality.

2.2. Defect Disposition

Every defect has a life cycle. Pictorially defect life cycle can be understood as below, Figure 2 [9]:

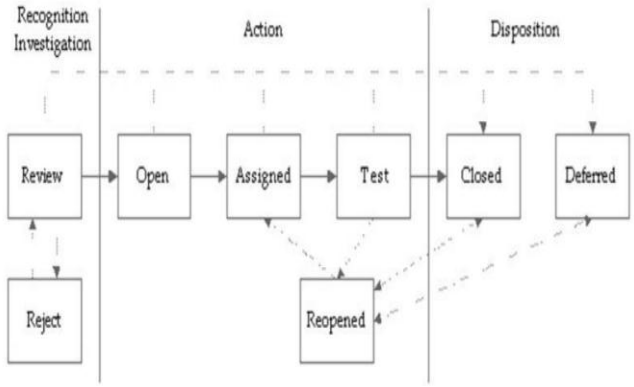


Fig. 2: Defect Lifecycle

A defect in its life cycle moves from one state to another – identification or recognition to final disposition. First stage is recognition when defect is first seen in terms of failures. Here defect is first seen and logged into defect tracking system. After defect is recognized it's investigated for root cause. Root cause only help to ensure what is causing failure and what could be probable solution. Solutions can be associated with 'direct root cause' or 'indirect root cause' [7]. Direct root cause is specific root cause of the problem and if it's fixed issue doesn't appear again. Indirect root causes even if fixed, issue might appear again in different events or actions or environments. Once investigation is done or root cause is found, associated actions are taken. Either it's decided to fix the defect or not. When these actions are taken, and concluded defect moves to disposition or terminal state.

All defects of different origins of Product X was analyzed and categorized into different defect disposition as part of this study. In Table 2 it can be seen which origin defects are associated with which disposition.

Table 2: Defect Origin Vs. Defect Disposition

Defect Origin		Defect Disposition								
May2013-April 2017	Teams	Defect fixed	Not a defect	Defect will not fix	Defect still not fixed	Defect existing (Duplicate)	Defect third party	Defect fix not known	Total Defect Disposition	Total
Developer	Core Engineering Team	2075	183	338	416	213	33	21	3279	3369
	Dev Automation Team	46	4	4	7	2	0	0	63	
	Tools Team	2	0	1	2	0	1	0	6	
	Application Developer Team	6	0	2	2	1	2	0	13	
	Build & Infra Team	4	0	0	1	0	0	0	5	
	Production Team	0	1	0	0	0	0	0	1	
	Research Team	0	1	0	0	0	0	0	1	
	White Box Testing Team	1	0	0	0	0	0	0	1	
Testing	Core Testing Team	6626	803	1413	1642	1199	308	103	12094	15030
	Test Automation Team	1118	71	161	205	204	17	2	1778	
	Test Certification Team	370	35	40	10	59	128	1	643	
	Driver Test Team	251	55	56	63	54	18	2	499	
	Hadware Test Team	7	3	2	1	0	1	0	14	
	Customer Customization Test Team	1	0	0	0	0	0	1	2	
Customer	End User [Non-Dev & Non-Testing]	737	113	224	272	249	66	66	1727	2395
	Customer [Outside Company X]	237	34	116	96	72	21	10	586	
	Marketing Team	48	2	10	12	5	4	1	82	
									Total	20794

There are total 7 different final state of defects are found:

I. Defect fixed – this is state where defect root cause is analyzed, found and fix was deployed in software build.

II. Not a defect – when analyzed it's found that these defects are not valid as either they are as per design or intended features are supposed to do the same. Customer or end users are not aware about design or feature or how to use the product.

III. Defect will not fix – such defects are deferred and not planned to be fixed. There might be multiple technical or business reason to do so. Fixing them will not bring value addition for the time being, so deferred to be fixed.

IV. Defect still not fixed – these defects are in open state and in active debugging to be fixed. These defects are planned to be fixed in upcoming milestone or releases on the product.

V. Defect existing (duplicate) – such defects when analyzed and root caused found that similar kind of defects already exist and tracked into defect database and being worked upon. Fixing existing defect will cause these defects to be fixed.

VI. Defect third party – there are many defects which are due to software components developed by other companies. Viz. Android OS related implementations, different Applications, games developed by another companies. They need to take actions to help fix the defect at their end. Company X Product X and for Release X – there is no action pending here.

VII. Defect fix not known – there are defects which are fixed due to design change or new features implemented, which are having similar root cause. Many changes coming in might fix few defects automatically and they disappear. Most of the time such defect root cause is found as "indirect root cause".

Most important aspects of Defects Disposition are 'defect fixed'. These defects are fixed because for these root cause was identified, and right corrective actions were taken. In Figure 3 we could understand that 4.91% customer bugs were really fixed as they were of high importance and causing bad user experiences or quality degradation. Out of almost 11.52% customer defects, only 4.91% defects are meaningful from disposition point of view.

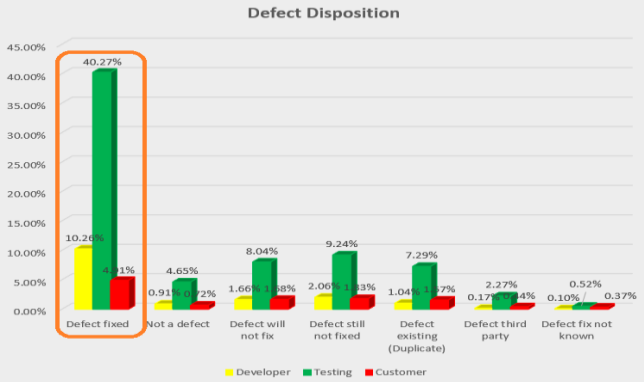


Fig. 3: Defect Disposition

2.3. Defect module

Based on this study Product X was analyzed and different major module or software area were defined. Total 10 major modules were defined. For sake of privacy of data these module names are not exposed here. These will be called Module 1, Module 2, and so on to Module 10 as in Table 3.

Table 3: Defect Origin vs. Defect Module

Defect Origin		Defect Module										
May2013-April 2017	Teams	Module 1	Module 2	Module 3	Module 4	Module 5	Module 6	Module 7	Module 8	Module 9	Module 10	Total DefectModule	Total
Developer	Core Engineering Team	0	21	33	0	649	235	294	1735	312	0	3279	3369
	Dev Automation Team	0	0	0	0	5	16	0	42	0	0	63	
	Tools Team	0	0	0	0	0	0	2	2	2	0	6	
	Application Developer Team	0	0	0	0	0	5	0	2	6	0	13	
	Build & Infra Team	0	0	0	0	0	4	0	1	0	0	5	
	Production Team	0	0	0	0	0	0	0	1	0	0	1	
	Research Team	0	0	0	0	0	0	0	1	0	0	1	
	White Box Testing Team	0	0	0	0	0	0	0	1	0	0	1	
Testing	Core Testing Team	0	27	49	0	1769	442	1907	3890	4010	0	12094	15030
	Test Automation Team	0	1	43	0	109	113	137	1307	68	0	1778	
	Test Certification Team	0	2	6	0	46	92	150	327	20	0	643	
	Driver Test Team	0	0	0	0	25	13	332	47	82	0	499	
	Hadware Test Team	0	0	1	0	1	2	5	4	1	0	14	
	Customer Customization Test Team	0	0	0	0	0	0	1	0	1	0	2	
Customer	End User [Non-Dev & Non-Testing]	0	0	3	0	247	102	268	721	386	0	1727	2395
	Customer [Outside Company X]	0	0	1	0	68	60	59	280	118	0	586	
	Marketing Team	0	3	0	0	10	10	11	21	27	0	82	
												Total	20794

Once these modules were defined, all defects were analyzed to see in which module they really belong to. In Table 3 there are 4 modules which contributed towardsmaximum defects and are having defects for more than 10% of each in Figure 4.

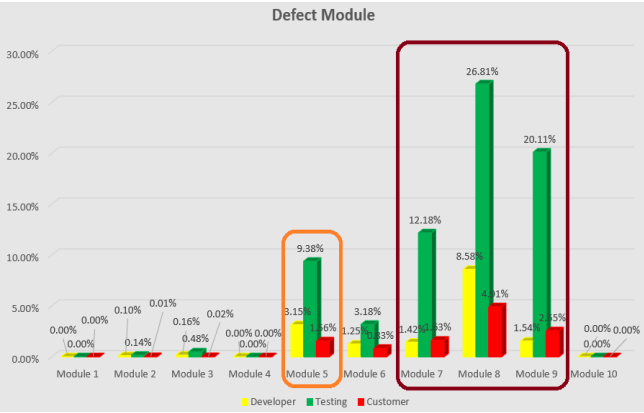


Fig. 4: Defect Module

Maximum customer defects are seen in module 8 (4.91%), module 9 (2.55%), module 7 (1.63%) and module 5 (1.56%). Based on this it was found that if these modules are given focus in terms of validation and based on root cause corrective actions, end user experience can be improved, and it will improve product quality eventually.

2.4. Test data

In Organization X whole test data based on Android based products were analyzed and put in Table 4. These tests data were analyzed based on: how many total tests are there in test database management system and are applicable for Android based system software products, how many of those tests were automated and manual. It was further analyzed based on different functional aspects and key performance indicator tests. Functional tests were categorized into different test types viz. requirement tests, interface tests, fault injection tests, resource usage tests, code coverage tests, boundary tests, certification tests, usability or out of box experience tests, design tests and product polish tests. Key performance indicator tests were further classified into performance tests, power tests and reliability tests. Reliability tests are further categorized into stress tests and stability tests [7]. These tests are mapped into one or another module defined and when these tests executed gives defects, which is mapped with defect severity. Based on this study it is corelated that defects origin, defects type/severity & test types could be mapped in Table 5. The kind of tests executed will mostly lead towards similar kind of defects which might have different severity and it will impact quality.

Table 5: Test Types Vs. Defect Types

Test Type	Defect Severity/Type
Requirement Tests	Functional [Functionality + Task Tracking + Enhancement + Corruption]
Interface Tests	
Fault Injection Tests	
Resource Usage Tests	
Code Coverage Tests	
Boundary Tests	
Certification Tests	
Usability/OOBE Tests	
Manufacturing Diagnostic (factory) Tests	
Design Test	
Product polish Tests	
Perf & Power Tests	Performance
Stress Tests	Application Crash
Stability Tests	System Crash

2.5. Root Cause of Defects

As part of this study all defects irrespective of origins were analyzed for root cause category [7]. In Table 6 all defects, their origins are mapped with root cause category could be seen.

Table 4: Test Types

Module	Total Tests (whole Test Database)	Automated Tests	Manual Tests	Funcational											KPI (Key Performance Indicator)			
															Perf & Power		Reliability	
				Requirement Tests	Interface Tests	Fault Injection Tests	Resource Usage Tests	Code Coverage Tests	Boundary Tests	Certification Tests	Usability/ OOBE Tests	Manufacturing Diagnostic (factory) Tests	Design Test	Product polish Tests	Performance Tests	Power Tests	Stability Tests	Stress Tests
Module 1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Module 2	435	222	213	367	0	0	0	0	0	0	14	0	0	0	42	12	0	0
Module 3	1457	649	808	1108	9	134	31	0	55	0	32	0	0	0	34	39	13	2
Module 4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Module 5	2256	470	1786	1246	11	47	38	0	35	0	162	3	18	0	507	68	39	82
Module 6	944	576	368	593	0	0	0	0	0	0	0	0	0	0	68	278	2	3
Module 7	4894	1605	3289	2839	263	39	0	0	5	0	1216	0	61	0	343	72	38	18
Module 8	8249	1541	6708	4290	1370	255	301	0	137	0	166	230	0	0	1255	163	20	62
Module 9	2508	19	2489	2238	0	0	0	0	0	0	117	0	0	0	130	19	0	4
Module 10	117	55	62	71	0	0	0	0	0	0	1	0	0	0	0	0	16	29
Total	**20860**	**5137**	**15723**	**12752**	**1653**	**475**	**370**	**0**	**232**	**0**	**1708**	**233**	**79**	**0**	**2379**	**651**	**128**	**200**

Table 6: Defect Origin vs. Root Cause

Defect Origin		Defect Root Cause Category											
May2013-April 2017	Teams	Test doesn't exist [tester level]	Test doesn't exist [customer HW specific]	Test doesn't exist [dev level]	Test exists [tests steps modification]	Test exists [execution missing]	Request for enhancement [RFE]	Duplicate of tester defect	Customer specific customization	Invalid [not actionable defect]	Defect reported by tester	Defect reported by developer	Total Root cause category	Total
Developer	Core Engineering Team	0	0	17	0	10	1140	4	0	396	0	1712	3279	
	Dev Automation Team	0	0	0	0	0	7	0	0	6	0	50	63	
	Tools Team	0	0	0	0	0	3	0	0	0	0	3	6	
	Application Developer Team	0	0	0	0	0	0	0	0	1	0	12	13	
	Build & Infra Team	0	0	0	0	0	0	0	0	0	0	5	5	
	Production Team	0	0	0	0	0	0	0	0	1	0	0	1	
	Research Team	0	0	0	0	0	0	0	0	1	0	0	1	
	White Box Testing Team	0	0	0	0	0	0	0	0	0	0	1	1	
Testing	Core Testing Team	10	0	0	0	5	1490	0	0	2002	8587	0	12094	
	Test Automation Team	0	0	0	0	1	461	0	0	275	1041	0	1778	
	Test Certification Team	0	0	0	0	6	21	0	0	94	522	0	643	
	Driver Test Team	2	0	0	0	3	12	0	0	109	373	0	499	
	Hadware Test Team	0	0	0	0	0	0	0	0	3	11	0	14	
	Customer Customization Test Team	0	0	0	0	0	1	0	0	0	1	0	2	
Customer	End User [Non-Dev & Non-Testing]	163	28	56	94	190	215	93	38	850	0	0	1727	
	Customer [Outside Company X]	41	7	18	14	41	44	29	56	336	0	0	586	
	Marketing Team	5	1	0	2	4	36	5	2	27	0	0	82	
													Total	20794

After analysis it was found that there are two categories of defects which are very crucial and would have been caught either if new tests were designed corresponding to that or existing tests would have been executed as part of release validation. For Product X and Release X, it's seen that 19.72% defects were invalid, 51.29%defects were already caught by testers; 8.57% defects were caught by developers. 1.88% defects were wishlist features from customers and so not of quality concern as these needs to be decided to be implemented in product or not. Two categories which cause quality to go down are: 1. Test doesn't exist (1.67%) – there were no tests in test database management which if would have been executed would have caught this or similar bugs. 2. Test exist (1.78%) – tests are there in test database but due to one or another reason were not executed for release X. In Figure 5, overall impact could be seen:

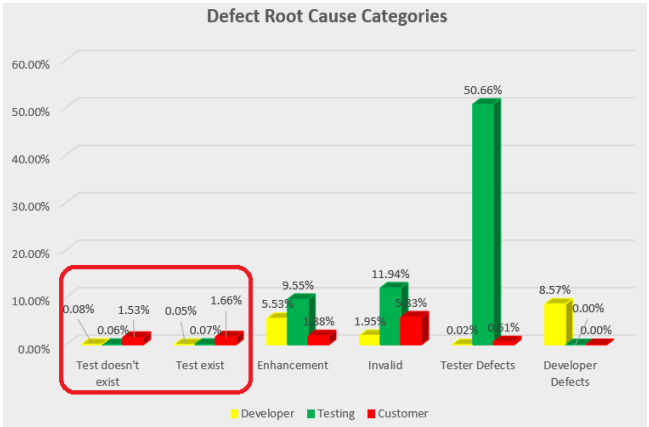


Fig. 5: Defect Root Cause Categories Distribution

2.6. Defect software stages

While this study all defects were analyzed that at which software stages they would have appeared based on root cause and fix provided for those defects. Software stages are taken based onstandard software engineering practices. For different origins – different team levels – which eventually leads to: developer, tester and customer level – all these aspects of defects were analyzed. In Table 7 consolidated defect data and at which software stages they map to could be seen. After further analysis to consider how it could impact quality it was seen that most of defects are caught at verification stage (64.40%) and implementation stage (17.69%). But at same time from customer defect perspective it's concluded that at implementation stage 5.66% defects would have been caught. At verification stage 3.69% defects were not caught and were reported by customers. These could also be aligned based on analysis made in Table 5 and Figure 5 – tests need to be designed at implementation stage and newly designed tests must be executed at verifications stage – based on root cause category "test doesn't exist, and test exist". Requirements gathering & analysis stage defects are aligned with root cause category "enhancement". This analysis is put in Figure 6:

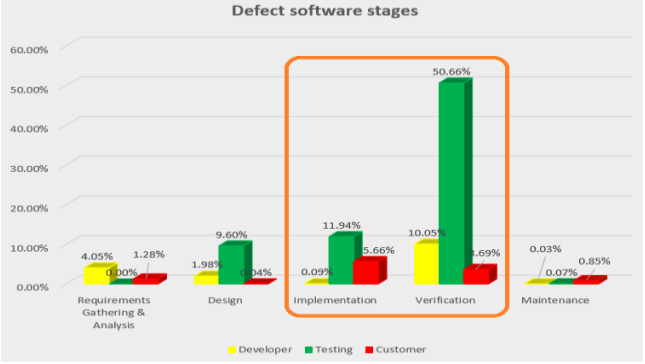


Fig. 6: Defect Software Stages

Table 7: Defect Origin Vs. Defect SW Stages

Defect Origin		Defect SW Stages						
May2013-April 2017	Teams	Requirements Gathering & Analysis	Design	Implementation	Verification	Maintenance	Total Module	Total
Developer	Core Engineering Team	834	409	19	2010	7	3279	3369
	Dev Automation Team	8	0	0	55	0	63	
	Tools Team	1	2	0	3	0	6	
	Application Developer Team	0	0	0	13	0	13	
	Build & Infra Team	0	0	0	5	0	5	
	Production Team	0	0	0	1	0	1	
	Research Team	0	0	0	1	0	1	
	White Box Testing Team	0	0	0	1	0	1	
Testing	Core Testing Team	0	1500	2002	8587	5	12094	15030
	Test Automation Team	0	461	275	1041	1	1778	
	Test Certification Team	0	21	94	522	6	643	
	Driver Test Team	0	14	109	373	3	499	
	Hadware Test Team	0	0	3	11	0	14	
	Customer Customization Test Team	0	1	0	1	0	2	
Customer	End User [Non-Dev & Non-Testing]	184	3	894	572	74	1727	2395
	Customer [Outside Company X]	70	3	226	185	102	586	
	Marketing Team	12	3	57	10	0	82	
							Total	20794

2.7. Software Quality Modeling

As part of study made in [6], existing software quality is merely defined as Pass %; which is: how many tests are executed and out of that how many tests are passed. Based on all attributes of defects, tests and software stages, it's seen that every defect originated impacts quality and it's has some quality weight associated. An algorithm was designed based on these attributes and overall weight associated with that. The skeleton or snippet of fully designed algorithm is as below:

/* Quality Algorithm – START */

/* Attributes under consideration as per dependency or flow – viz. defect origin leads to defect severity & so on… */

/* Different variables */

1. Initial Quality [Pass %]
2. Defect Origin
.
.
13. Weight (Module Vs. Test)
14. New Quality [New Pass %]

/* Pass% is dependent variable */

/* Rest all variables are independent variables which will impact Pass% */

/* Weight of attributes*/
1. If Customer then{
1.a Weight (Customer all);}
2. If Customer AND [Functional OR Stress OR Stability OR Perf & Power] then {
2.a Weight (Functional);
2.b Weight (Stress);
2.c Weight (Stability);
2.d Weight (Perf & Power);}
3. If Customer
AND [Functional OR Stress OR Stability OR Perf & Power]
AND [Regression OR Not a Regression] then {
3.a Weight (Regression);
3.b Weight (Not a Regression);}
/* Same way weight is calculated for other variables viz. Defect priority, defect disposition, defect root cause,defects SW stages and defect module – captures: 4th, 5th, 6th& 7th stages of calculations */

8. If Customer
AND [Functional OR Stress OR Stability OR Perf & Power] AND [Regression OR Not a Regression]
AND [P1 OR P2 OR P3 OR P4] AND [Valid AND Closed]
AND [Test doesn't exist, OR Test exist OR Enhancement]
AND [Requirements OR Design OR Implementation OR Verification OR Maintenance] AND [M(I) – I = 1 to N] then
/* in this study N = 10 */
{
8.a Weight(M1);
.
.
8.j Weight(M10);}

/* Every customer defect is mapped with a module based on different attributes and defect attributes are connected & dependent */
/* Weight of each defect associated with module is now know - calculate total quality of module */

Calculate Module Overall Weight [M(I)]
{If PRODUCT [M(I) where I = 1 to 10] = 0 then
"Ignore that attributes weight in calculation"
Else /* it's non-zero */
Calculate Weight M (I);
Average of [Weight M (I);}

/* Defect Module and Test Module towards Test Types and it's weight */
9. Calculate weight of "Test Module"
10. Calculate weight of "Test Type" based on "Defect Severity" weight
11. If Defect Origin = "Tester OR Developer" then
/* test cases based defects are mainly originated from Tester/Developer */
AND Module M(I) – I = 1 to N
/* Here N = 10 */
AND Module Test Type [Functional OR Stress OR Stability OR Perf & Power]
AND Defect Severity [Functional OR Stress Stability OR Perf & Power]
AND Defect Disposition [Valid] then {
11.a Weight(Module);}

/* Every module is mapped now with test attributes */ /* TM = Test Module */

Calculate Test Module Overall Weight TM(I)
{If PRODUCT [TM(I) where I = 1 to 10] = 0 then
"Ignore those attributes weight in calculation"
Else Calculate Weight TM(I);}
For every module M(I) I = 1 to N {
Module Initial Quality = M;
Module New Pass % = M - [(M(I) + TM (I)];}

/* Quality Algorithm – END */

2.8. Existing Quality

For product under study "Product X" and release made "Release Y" was again evaluated for quality based on this new algorithm. Existing quality model was only based on number of tests executed and how many tests were passed [6].
The algorithm designed to calculate the quality was based on attributes part of study and on same dataset it was executed to see what is quality for Product X and Release X. Whole dataset of defect and test are kept in Microsoft Excel and based on attributes identified and defined, different macros and formulas were developed to operate on those data. The algorithm was at this moment implemented in terms of Excel available features for weight calculation.
After calculating weight of all attributes against total defects and total tests it's found that test passed are not the right representation of quality in its current state. Table 8 shows that based on traditional quality model Pass% was 93.07% (which was acceptable to release as per Pass % Legend) but that brought many customer defects which caused bad experience or quality. When all these taken into consideration and different aspects of defects, tests and their dependencies were analyzed as part of this study, the same product and same release gave Pass % as 83.92% (which is not acceptable – as per color coding of Pass% Legend) – based on new quality model & quality algorithm.

Table 8: Existing Quality Vs. New Quality

Module	Release X of Product X- Tests Quality Status			Product X [Release X]		
	Total Tests (X)	Target Pass %	Old Pass % [Release X]	Weight (Customer Defect)	Weight (Test Type [Tester Defect AND Developer defect])	New Pass % [Release X]
Module 1	0	NA	NA	NA	NA	NA
Module 2	435	NA	NA		NA	NA
Module 3	1457	95%	60.00	0.38	0.75	58.87
Module 4	0	NA	NA	NA	NA	NA
Module 5	2256	95%	99.74	0.31	1.07	98.36
Module 6	944	95%	99.59	0.31	0.93	98.34
Module 7	4894	95%	93.26	0.30	1.56	91.40
Module 8	8249	95%	92.51	0.26	2.07	90.17
Module 9	2508	95%	87.18	0.26	0.93	85.99
Module 10	117	95%	NA	NA	NA	NA
Total	20860	95%	93.07	1.83	7.32	83.92
Pass % Legend	< 90%	>= 90% & < 95%	>= 95%	Conclusion: Actual Quality is lower than what Pass% says		

Based on this quality algorithm two levels of weight are calculated:
1. Based on "Customer Defects" – mapped to "Test Modules".
2. Based on "Test Types" – mapped to "Defect Severity" and eventually mapped to "Test Modules".

3. Conclusion and Future Work

In this study a "Quality Algorithm" is designed which tells the quality of Android based software system in its current state. Based on this study and previous studies made [6 & 7] it can be concluded that:

1. Software quality is not just how many tests are executed and out of those how many tests are passed.
2. There are other attributes who plays active role in defining software quality.
3. "Defects & Tests" – and its different attributes play major role in depicting the real quality.
4. Defects and its attributes: Defect Origin, Defect Severity, Defect Regression, Defect Priority, Defect Disposition, Defect Root Cause, Defect Corrective Actions, Defect Software Stages & Defect/Test Modules – these are attributes which are interdependent and impacts quality. These all help define "Defect Weight" which impacts quality.
5. Tests and its attribute: Test Type is very important aspect which impacts quality. Test Type in relation with Defect Severity, Defect Disposition & Test/Defect Modules – are those attributes which defines "Test Weight" which impacts quality.
6. When all attributes taken into consideration; current quality is more accurate. Viz. for Product X, Release X: Initial Pass% was 93.07% but when calculated based on defect & test weight New Pass% is 83.92%. It's because nature of defects and tests can't be ignored for quality representations.
7. Release X was made to customers based on initial quality model but based on new quality model – it's not acceptable to release as Pass% is not as per release milestone.

3.1. Future Study Focus

1. Take Release (X+N) and validate the new algorithmic quality model defined. Validate whether the "Root cause & Corrective actions" defined when implemented in "Product X" is helping to improve quality for future releases.
2. Validate the new algorithmic model on different product(s) of Company X.
3. Study quality model of a different company and analyze attributes and corelate if model is same or different.

4. Limitation

1. This "Quality Algorithm" is only validated by running on Product X & Release X – which is part of study. Based on different "Root cause & Corrective actions" attributes and after it's implementations, need to validate the same algorithm on next releases of Product X Viz. Release (X + N). Another level of validation must be done on different products of Company X too.
2. While calculating weight for different attributes were not taken into consideration which are as below:

- Defect Origin (External vs. Internal customers). All customer defects are here based on "End User, Marketing Team and Customer". End user and Marketing team is within Company X only who are non-tester and non-developer so considered as customers. In future study need to validate how these separately impacts quality.
- Defect Regression – here only two attributes were considered as part of study. One is based on defects categorized into regression and other as not a regression. Few defects are there which were not falling in either of these two and are called "Undetermined Regression". They were not analyzed for creating weight and impact on Quality.
- Defect Priority – here only four attributes were considered for calculating weight and impact. These are: release blockers, fix before next build, fix before next release and nice to have. The one which are not taken into consideration are defects which are not prioritized (U – Unprioritized). No priority defined means these might not fall into any of above four, so not taken as part of this study.
- Defect Disposition – defects which were valid and closed are only ones which were analyzed. There are defects which were invalid or decided not to be fixed or are in open state; they are not considered for weight calculation as they are of least importance from software management team and quality point of view.
- Defect Root Cause Category – here defects which are valid and corresponding actions either "test exist" or "test doesn't exist", are only considered for calculating weight. Defects which are invalid, enhancement, already raised by tester or developer were not taken into consideration. It's based on assumption that "there must at least be a test case which can catch customer defect".
- Test type – for this attribute only corresponding "Defects Severity" is taken into study. For those defects only if disposition is Valid; they are only taken into consideration for test weight calculation.

3. Code coverage aspects are not analyzed. Viz. are there tests which are capable to execute all lines or functions of the software product. This will be mainly associated with two major attributes

of root cause; “test exist or doesn’t exist” to execute missing lines of code.
In future study all above limitations can be taken into consideration to analyze the impact of software system quality.

5. Acknowledgement

A sincere thanks to Prof. (Dr.) R. Kamatchi, Professor, Head, CSE, Amity School of Engineering and Technology, Mumbai; being mentor and guide and for continuous help and support to come up this paper.

6. References

[1] J.M.Juran, "Juran's Quality Control Handbook",McGraw-Hill, (1988).
[2] International Organization for Standardization, "ISO/IEC9001: Quality management systems -- Requirements" (1999) and International Organization for Standardization, "ISO/IEC 24765: Systems and software engineering – Vocabulary," (2010).
[3] W. A. Shewhart, Economic control of quality of manufactured product. Van Nostrand, (1931).
[4] A. V. Feigenbaum, "Total Quality Control", McGraw-Hill, (1983).
[5] Marek Leszak; Lucent Technologies, Optical Networking Group, Thurn-und-Taxis-Str. 10, 90411 Nuernberg, Germany&Dewayne E. Perry; Electrical and Computer Engineering, The University of Texas at Austin, Austin, TX. Dieter Stoll; Lucent Technologies, Optical Networking Group, Thurn-und-Taxis-Str. 10, 90411 Nuernberg, Germany(2000), “A case study in root cause defect analysis Proceeding”. ICSE '00 Proceedings of the 22nd international conference on Software Engineering Pages 428-437 ACM New York, NY, USA ©2000 table of contents ISBN:1-58113-206-9 doi>10.1145/337180.337232
[6] Abhishek Anurag&Kamatchi Iyer, Computer Science & Engineering, Amity School of Engineering & Technology, Amity University Mumbai (2018) “A Case Study of existing Quality Model based on Defects & Tests Management of Embedded Software System” in International Journal of Computer Science Engineering and Information Technology Research (IJCSEITR) journal www.tjprc.orgwith ISSN (Print): 2249-6831; ISSN (Online): 2249-7943; Impact Factor (JCC): 8.9034; Index Copernicus Value(ICV): 60.28 NAAS Rating : 3.76; IBI Factor : 3.2. Published: Apr 30, 2018; Paper Id.: IJCSEITRJUN20183. http://www.tjprc.org/viewarchives.php?keyword=Abhishek+Anurag&jtype=2&from_date=&to_date=&journal=14
[7] Abhishek Anurag&Kamatchi Iyer, Computer Science & Engineering, Amity School of Engineering & Technology, Amity University Mumbai (2018), “Study on Software Quality improvement based on Root Cause & Corrective Actions” (Paper-id: ICETCT181055). International Journal of Advanced in Management, Technology and Engineering Sciences, IJAMTES/319, ISSN No.: 2249-7455, IJAMTES Journal, Volume 8, Issue IV, April – 2018.
[8] Yehudai, Amiram; Tyszberowicz, Shmuel&Nir, Dor (2007). “*Locating Regression Bugs*”. Haifa Verification Conferencehttps://en.wikipedia.org/wiki/Software_regression;eBook@ https://www.researchgate.net/publication/225437428_Using_Virtual_Coverage_to_Hit_Hard-To-Reach_Events#page=235;. Retrieved 10 March 2018.
[9] Rex Black, President, RBCS, Inc. (2015) “The Defect Life Cycle and the Software Development Life Cycle”, https://rbcs-us.com/site/assets/files/1116/the-defect-life-cycle-and-the-software-development-life-cycle.pdf Copyright © 2015, RBCS, All Rights Reserved.