A STUDY OF COLLECTING CUSTOMER NEEDS IN SOFTWARE DEVELOPMENT PROCESS AND ITS IMPACT ON BUSINESS OF SELECTED IT COMPANIES IN PUNE

A Thesis Submitted to

Tilak Maharasthra Vidyapeeth, Pune

For The Degree of Doctor of Philosophy (Ph.D.)

In Management Subject

Under The Board of Management Studies

Submitted By

Mrs. Ashvini Pradeep Shende

Under The Guidance of

Dr. Prasanna G. Deshmukh

September 2017

Declaration by the Candidate

I hereby declare that the thesis entitled **"A Study of Collecting Customer Needs In Software Development Process and Its Impact on Business of Selected IT Companies In Pune"** completed and written by me has not previously formed the basis for the award of any Degree or other similar title upon me of this or any other University or examining body.

I further declare that the material obtained from other sources has been acknowledged in the thesis.

Mrs. Ashvini P. Shende

(Research Student)

Place: Pune

Date: 12 September 2017

Certificate of the Guide

This is to certify that the thesis entitled **"A Study of Collecting Customer Needs In Software Development Process and Its Impact on Business of Selected IT Companies In Pune"** which is being submitted herewith for the award of the Degree of Philosophy (Ph.D.) under the faculty of Management of Tilak Maharashtra Vidyapeeth, Pune, is the result of original research work completed by Mrs. Ashvini Pradeep Shende, under my supervision and guidance.

To the best of my knowledge and belief the work incorporated in this thesis has not formed the basis for the award of any Degree or similar title of this or any other University or examining body upon her.

Dr. Prasanna G. Deshmukh

(Research Guide)

Place: Pune

Date: 12 September 2017

Acknowledgement

I owe my personal thanks to **Dr. Sanjay Kaptan** (Principal, Indira College of Commerce and Science), **Dr.Janardan Pawar** (Vice-Principal, Indira College of Commerce and Science), **Dr.Anjali Kalkar** (Vice-Principal, Indira College of Commerce and Science) for their whole hearted support, motivation and inspiring encouragement.

I am extremely grateful to my guide **Dr. Prasanna G. Deshmukh**, **Principal**, Anantrao Thopate Arts, Science and Commerce College, Pune, for his encouragement and continuing support in this endeavor. His deep insights helped me at various stages of my research. Under his expert guidance I have completed my doctoral research on the topic **"A Study of Collecting Customer Needs in Software Development Process and Its Impact on Business of Selected IT Companies in Pune"**

Very special thanks to **Dr. Deepak J. Tilak** (Vice-Chancellor, Tilak Maharashtra Vidyapeeth, Pune) and **Dr. Sunanda Yadav** (HOD, Ph.D Section, Tilak Maharashtra Vidyapeeth, Pune) for giving me the opportunity to carry out my doctoral research.

My sincere gratitude and heartfelt thanks is reserved for **Sarita Byagar, Shivendu Bhushan, Thomsan Varghese, Sarika Thakare, Vijaya Kumbhar, Manisha Patil, Mahesh Jagtap, Dr.Suresh Pathare, Avinash Shingate**, **Shantilal Ghalme** and my all colleagues and **Non-Teaching Staff** for their timely guidance and valuable inputs. A big "Thank you!" also goes out to all IT companies **Project Managers and Software Engineers Mr.Sagar Thombre**, Mr.Somen Parmanik, Mr. Sharif Malik, Mr. Prabhakarn K., Mr. Hanumant U., Mr.Aniket Pawar, Mr. Mandeep Singh,Mr.Ronak C. Mr.Saqub A. Mr.Jayesh S. Mr.Rohit Thakur, Mr.Datta Pawar who helped me by giving the required information. I also want to mention **Dr.Manisha Patil**, **Project Manager**, Nice Systems, Pune and all my students for their support.

Finally, I would like to acknowledge the most important people in my life my husband **Mr. Pradeep Shende,** my parents **Ms. Anita Lagad,** my in-laws **Mr. Pramod Shende** and **Mrs. Mandakini Shende** & my son **Sharvil** for their extraordinary support, love patience and understanding throughout my long journey**.** Their support and tolerance has proved great help in the fulfillment of this mammoth task.

Mrs. Ashvini Pradeep Shende

C O N T E N T S

List of Figures

Chapter 1 Introduction

Chapter 2 Research Methodology

Chapter 3 Review of Literature

Chapter 5 Conclusions and Suggestions

List of Tables

Chapter 2 Research Design and Methodology

Chapter 4 Data Presentation, Analysis and Interpretation

List of Graphs

ABBREVIATIONS

SOP-: Statement of Purpose

PMC-: Pune Municipal Corporation

PCMC-: Pune Chichwad Municipal Corporation

RA Engine-: Requirement Analysis Engine

CNMRSF-: Customer's Needs Management to Reduce Software Failures

ABSTRACT

On

A STUDY OF COLLECTING CUSTOMER NEEDS IN SOFTWARE DEVELOPMENT PROCESS AND ITS IMPACT ON BUSINESS OF SELECTED IT COMPANIES IN PUNE

1. INTRODUCTION

In today's IT world, more and more online Applications and tools are used, which help in carrying out various daily chores. If these applications and tools do not work according to specification then it would cause inconvenience to all users. But it has almost been a too many decades since the software industry has detonated. It has witnessed a remarkable growth and a tremendous growth not only in the core activities but also in the IT enabled services. Despite the uninterrupted expansion, the software industry still has the highest number of project delays and failures. According to the Standish report, 44% of the software projects are challenged (late, over budget and/or with less than the required features and functions) and 24% have failed (cancelled prior to completion or delivered and never used). Thus, making a total of 68% (both challenged and failed) which is quite exponential. Boehm found that 15-35% of all the software projects were cancelled outright while the remaining projects suffered either from schedule slippage, cost overruns or failure to meet the project goals. [1]

Miscommunication between requirement management team and development and testing team is one of the major reasons for the project failure. And root cause of this miscommunication between requirement management team and other teams of the software department is the volatile nature of collected needs coming from end customers or clients. Collected needs in volatile nature are by default because we cannot skip the changing nature of Customer's demands. And here software programmer and testing becomes slave to fulfill the volatile collected needs from the client. [2]

C. J. Davis, Fuller, Tremblay, & Berndt found accurately capturing system collected needs is the major factor in the failure of 90% of large software projects," echoing earlier work by Lindquist who concluded "poor collected needs management can be attributed to 71 percent of software projects that fail.[25]

Collected needs play a driving role during the product creation because in every software development method, requirement gathering and analysis phase plays the most important role. Stability of collected needs potentially makes an impact on the success of later phases in a software project, including the success of test cases. According to Brooks, the toughest part in building a software system is to decide precisely what needs to be created. Furthermore, the poor requirement gathering and analysis may affect negatively at a later stage. Moreover, predicting potential results of the later phases from early time of software development can obviously help the project team to better deal with the risks of project rescheduling and resulting in a low-quality product. [3]

The success of software project depends on the quality of collected needs specification. Even though we have good collected needs specification in the beginning, there will be collected needs changes during the project development which may have the impact on testing process.

Chapter Scheme

Chapter 1. Introduction

Chapter 2. Research Methodology

Chapter 3. Review of Literature

Chapter 4. Data Analysis and Interpretation.

Chapter 5. Conclusion and Suggestions.

Chapter 2. Research Methodology

In this chapter the method of selection of the sample is described in this chapter and after that the nature of primary data and secondary data is explained.

The researcher has used survey based research methodology to carry out this research. The study is related to verify the impact of poor requirement gathering analysis on software testing.

The researcher has considered the Pune and PCMC area for the study. This study is primarily focused on awareness of various tools used during development of software and problems face by testers in software companies in Pune and PCMC.s that is why primary data was collected from employees of Software Company in Pune. Researcher has used interview and questionnaire data collection method. Researcher has collected data from software companies from Hinjewadi, Magarpatta (Hadapsar) ,Shivaji Nagar and Kharadi.

By applying purposive sampling, Total 21 companies have been identified for study which has more than 250 cr. Turnover. [93-98]

Type of Industry	**Total Companies**	**5 % Sample of Companies**
Software Companies	424	21

Table No. 1. Software Companies

By applying Quota sampling, Researcher has divided respondents in 3 categories Business Analyst, Designer, Testers.

Sr. No.	Company Name	No. of Employees
1	Accenture	7
2	Amdocs	24
3	Atos	46
4	Davachi	7
5	BMC	6
6	Capegemini	53
7	Citi Bank	6
8	Congnizant	9
9	Hummingbird	9
10	Calsoft	16
11	Neptune InfoTech	2
12	IBM	4
13	KPIT Cummins	38
14	Patni	7
15	Persistent	9
16	Principal Optima	4
17	CLSA	7
18	Sigma Soft	51
19	Symphony	51
20	Tech Mahindra	34
21	Wipro	10

Table No.2. Distribution of

Employees in different software companies

Type of Respondent	No of Employee
Business Analyst	134
Designer	119
Tester	147
Total	400

Table No.3. Occupation of Employees

As per Krejcie and Morgan's law(1970) if population is in between 75,000 and upto 10,00,000 then 384 sample size should considered so here in research researcher has considered it as 400.

Researcher has used statistical package for the social science (IBM SPSS 20) to test the hypothesis and analysis of the data.

Pilot Study

The researcher has conducted pilot survey randomly to test the questionnaire. To know the consistency of questionnaire to be administered for the research, researcher has applied the Cronbach's Alpha reliability test. Initially the questionnaire was circulated to 5 Software Companies from which 47 Business Analyst responded and reliability test was conducted.

The result of pilot survey about Business Analyst is given in following Table.

		No. of Respondents	%	Cronbach's Alpha	N of Items
Cases	Valid	47	100.0	.713	82
	Excluded[a]	0	0.0		
	Total	47	100.0		

Table 4. Reliability Statistics of Pilot Survey of Business Analyst of Software Companies.

It is observed that questionnaire is consistence and Cronbach's Alpha score is 0.713. It means 70 percent respondents understood the questionnaire. Thus researcher concludes that this questionnaire can be administered for the further research.

Secondary Data

Various sources like journals, government reports, Ph.D. Thesis, books, magazines, and internet are explored to collect secondary data and same has been used to support the objectives and hypotheses whenever it needed.

A- STATEMENT OF THE PROBLEM

A good set of collected needs are the base for any software project. Requirement gathering phase is playing main role to estimate cost and schedule as well as developing design and testing specifications. Hence quality of collected needs playing main role in the success of any software project.

- According to the **Standish report**, 44% of the software projects are challenged (late, over budget and/or with less than the required features and functions) and 24% have failed (cancelled prior to completion or delivered and never used). Thus, making a total of 68% (both challenged and failed) which is quite exponential. Boehm found that 15-35% of all the software projects were cancelled outright while the remaining projects suffered either from schedule slippage, cost overruns or failure to meet the project goals. [1]
- Even though collected needs are freeze in initial phase of software project but it may get changed throughout the software development lifecycle.

- Change in collected needs means it can be addition, deletion or modification. Such kind of change in requirement during SDLC always impacts the cost, schedule and quality of software product.
- The reason to fail any software product is mainly depends upon the quality of collected needs. Hence, a good set of user collected needs are needed for any software project, to be successful. But if collected needs are not specified clearly, correctly against what the system should do, then many projects will fail in this case. In fact, many systems have just been given a deadline for delivery, a budget to spend, and a vague notion of what it should do.

  As a result, testing phase faces many problems during implementation of software.

B- SCOPE OF THE STUDY

The study is related to the analysis of impact of poor requirement analysis and gathering process on software testing. Pune city has been considered for this research work.

The scopes of this research are software companies in Pune.

During the course of the present study the researcher has focused on the study of the impact of poor requirement gathering process on software testing. Also it is focused on provision of model which will help to reduce the failures of software product due to poor requirement gathering process. This model has been designed by considering parameters like cost, time etc. The researcher has also done analysis of current scenarios of software testing and tools used in software industries.

C- OBJECTIVES OF THE STUDY

The main objective is to study difficulties of Software Testing and to find root cause of software failures. The study has following detailed objectives:

1. To study various task undertaken for software development process in IT Companies.
2. To Study the various tools and techniques used in collecting initial needs for the software product development.

3. To identify various factors responsible for the software development.
4. To study impact of Collecting needs from customer on business of IT companies.
5. To draw conclusion and suggestions.

D- HYPOTHESIS OF THE STUDY

In consistent with the objectives, the researcher formed following hypotheses:

Hypothesis 1: There are hurdles in collecting customer needs in software development.

Hypothesis 2: IT Industry follows standard practices to use licensed or well-known tools to collect initial needs from customer in software development.

Hypothesis 3: If collected needs are not freeze, then it has impact on business.

Testing of Hypothesis

In this research four hypotheses were stated, these entire four hypotheses are tested using SPSS statistics 20 tool, and applied test.

Hypothesis 1-: KMO and Bartlett's Test

Hypothesis 2-: Z statistics test

Hypothesis 3-: Z statistics test

Hypothesis 1: There are hurdles in collecting needs process for software development

Kind of Customer's needs	Yes	No	Percent(Yes)
Scope Clarification for Domain	21	379	5.3
Input Processes	7	393	1.8
Reporting Procedures	44	356	11.0
Number of Users	7	393	1.8
Data Collection	12	388	3
All of the Above	309	91	77.3

Methods for collecting requirement (The above Table is with reference to same chapter Table no. 4.7).

Test Statistic: – KMO and Bartlett's Test

H0: Null Hypothesis: There are no hurdles in collecting needs process.

H1: Alternate Hypothesis: There are hurdles in collecting needs process.

This hypothesis has been tested by Business Analyst and developers; they mentioned various hurdles in collecting needs process using KMO and Bartlett's Test. To study the hurdles face by developers, factor analysis is used to develop concise multiple item scales for measuring

various constructs. This test is carried out by using Barletts test of Sphericity which checks the determinant of correlation matrix into consideration which converts it into a chi-square statistics. Another condition needs to be fulfilled before factor analysis would be carried out Kaiser –Meyer-Olkin (KMO) statistics.

Kaiser-Meyer-Olkin Measure of Sampling Adequacy.		.515
Bartlett's Test of Sphericity	Approx. Chi-Square	1549.871
	Df	15
	Sig.	.000

Table No. 5 KMO and Bartlett's Test

From the survey and data analysis it has been seen that scope clarification for domain, input processes, reporting procedures, number of users and data collection are all the methods are most important and suitable for error free requirement collection. Using all these types of customer's needs results into error free requirement specification, this in turn results into error free software project.

Hypothesis 2: IT Industry follows standard practices to use licensed or well-known tools to collect initial needs from customer in software development.

Collecting needs Tools	Yes	No
Visual Paradigms	336	64
Project Management Software	238	167
Microsoft-Package	235	165
Data Dictionary	166	234
Use Cases and User Stories	160	240
ReqHarbor.com	357	43
MindTool	221	179
IBM Rational Doors	379	21
Jira	146	254
Rally	124	276
Taleo	150	250
Quality Center	318	82

Collecting needs Tools (The above Table is with reference Chapter 4Table no. 4.17).

Step 1: Setting Hypothesis

H0: 95% or more employees agreed that it is best practice to use collecting needs tools used in IT Industry. (H0: p = .95)

H1: < 95% or more employees agreed that it is best practice to use collecting needs tools used in IT Industry. (H0: p = .95)

(H1= p < .95)

H0 : p =0.95

H1= p < 0.95 (One tail test as rejection area is towards one side)

Step II: Sample Size

n=400 (> 30) As n > 30, large sample test i.e. Z-test is used.

Step III: Calculation of S.E. (Standard Error)

$S.E = \sqrt{pq/n}$ Where p = 95

q = 100-p = 5

$S.E. = \sqrt{95*5 / 400} = 0.2179$

Step IV: Calculation of Z value.

Z= diff. / S.E. diff = 95-94.75

Zcal= 1.1473

Step V: Comparison:

Table value of Z for one tail test at 5% level of significance is 1.64

Step VI: Conclusion:

Calculated value of Z (1.1473) < Table value of Z (1.64) Hence we accept H0 which means 95 percent System Analyst have a positive attitude towards usage of tools for Collecting needs in IT Industry and hence the hypothesis of the study is accepted.

Hypothesis 3: If collected needs are not freezed, then it has impact on business.

Referring above table 4.29 following hypothesis is proved

H0 -: On an average, clients are taking 35% of time duration for SOP (i.e $\mu = 0.35$)

H1-: On an average, clients are taking more than 35% of time duration for SOP, which has an impact on business (i.e $\mu > 0.35$)

Calculation of Z value.

Sample mean = 0.3893

Population mean under H0 is $\mu = 0.35$

Z= diff. / S.E.

where,

diff = 0.3893 – 0.35 = 0.0393

S.E. = $\sigma/\sqrt{n} = 0.009202$

Zcal = 0.0393 / 0.009202 = 4.2738

Table value of Z for one tail test at 5% level of significance is Ztab = 1.64

Conclusion:

Since Calculated value of Z (4.2738) > Table value of Z (1.64)Hence, we accept H1 which means that, on an average, clients are taking more than 35% of time duration for SOP, which has an impact on business.

Chapter 3 LITERATURE REVIEW

In this chapter, Researcher found numerous research studies pertaining to analyzing software requirement and testing process in software industry. The researcher has done a literature review on various aspects of software development like impact assessment of software collected needs on development process, testing process in software industry, programmers perspective of gathering collected needs, comparative study of development

process models and future prospects for developing error free software product. Researcher has defined 23 Characteristics, which are further used for designing questionnaires.

Following are the Important Characteristics of requirement analysis and software testing process

- kind of collected needs
- Users Involved in Requirement Analysis
- Awareness of following Software Development Life Cycle process
- Techniques use to gather software requirement
- significance of various documents in success of software project
- Factors reasonable for the failure of software project
- Factors contributing to failure for requirement gathering process
- Testers plays a role beginning of SDLC
- Tester role present in Requirement Phase
- Tools used in gathering collected needs Process
- Testing type i.e Manual or Automated
- Awareness of testing tools
- Various tools used for software testing
- Test cases execution
- Factors responsible for change in requirement gathering may affect on software testing process
- Factors responsible to make software erroneous
- Defects are raised on incorrect requirement
- Overhead occurs in software testing due to poor requirement gathering
- Common requirement issues that may affect Software Testing
- Document is most useful in success of software testing
- Different cost that are considered during testing process
- Occupation of employees in software Industry

Most of the studies reveal requirement is base for success of software and how wrong collected needs impact on testing process. Some study are also considered as per employees' point of view usage of requirement gathering tools and testing tools benefited for error free software.

Chapter 4. Data Analysis and Interpretation.

A- ANALYSIS

The fifth chapter analyzes the data obtained from Developers and testers of Software Companies. The analysis is carried out under various titles after doing pilot survey which is as under:

In this research, 400 respondent's data collected from Business Analyst, Designer and Tester from different software companies reside under PMC and PCMC area.

The data for employees of Software companies has been collected through interviews & questionnaires then compiled in 32 tables.

Researcher has done analysis for following points.

- **Employees from Software Companies:** In this research, 400 respondent's data are collected from Business Analyst, Designer and Tester from different software companies they reside under PMC and PCMC area.
- **Distribution of Business Analyst, Designer and Tester from different software companies present under PMC:** 400 respondents data are collected from different software companies resides in PMC and PCMC area. Among 400 respondents, 190 employees belong to software companies reside in PMC area and 210 employees belong to software companies reside in PCMC area.
- **Gender and Occupation wise Distribution of Employees:** For this research, this analysis has been done based on gender of employees. From the analysis, it has been seen that 28.5 percent male employees and 5 percent female employees have designation as business analyst and 22.5 percent male and 7.5 percent female employees having designation as Designer. Researcher also collected data for 22.5 percent male and 14.5 percent females employees are working as a Tester. From this analysis, it has been

proved that employees are working as Business Analyst, Designer, and tester in different software companies.

- **Qualification and Occupation wise Distribution of Employees:** For the understanding and awareness about requirement gathering process, researcher has collected educational wise data from 21 of different software companies. For this research, this analysis has been done based on qualification of employees. From the analysis, it has been seen that 28 percent Postgraduate and 5.5 percent graduate employees are Business Analyst. 23.75 percent postgraduate and 6 percent graduate employees is Designer and 21 percent postgraduate and 15.75 percent graduate employees are Testers. From this analysis, it has been proved that employees are graduate, postgraduate qualified, and they are having knowledge about requirement gathering process with designation wise experience.

- **Distribution of Business Analyst, Designer and Tester from different software companies present under PMC and PCMC area:** As we have considered PMC and PCMC areas from Pune city, Hadapsar, Kharadi and Shivaji nagar office locations belongs to PMC area and Hinjewadi belongs to PCMC area. Hence from the data analysis, it has been proved that 49 employees are from Software companies which are located in Hadapsar area, 210 employees from Hinjewadi area, 36 from Kharadi and 105 from Shivaji Nagar.

- **Requirement Gathering Techniques used in Software Industry:** From the analysis of respondent views, it has been shown that, Personally Meeting with client is the best requirement gathering technique as it is nothing but face to face communication with client and using this technique business analyst can easily get clarified all the doubts regarding requirement specification. Hence, maximum around 369 respondents i.e 92.3 percent provided positive vote for "Personal Meeting" requirement gathering technique and this information supports the ***First Objective*** of the study.
- **Useful tools for requirement gathering process:** From the survey, it has been seen that 379 respondents are agreeing that 'IBM Rational Doors' is the best requirement gathering tool. Because 'IBM Rational Doors' supports multiple functionalities for collected needs

like Requirement gathering, Software Design, Task Management and Collaborative Modeling and this information supports the ***First Objective*** of the study.

- **Types of collected needs need to use for betterment of software project :** From the survey and data analysis it has been seen that scope clarification for domain, input processes, reporting procedures, number of users and data collection are all the methods are most important and suitable for error free requirement collection. Using all these types of collected needs results into error free requirement specification, which in turn results into error free software project.

- **Time Duration for Client interaction while gathering collected needs:** To collect error free requirement from client, there is need to consume adequate time for requirement gathering process. However, 50 percentage employees are voting to 4 weeks time duration for requirement gathering process.

- **Useful traps for requirement gathering process:** To avoid this incorrect and incomplete requirement issue, there is need to user few traps these traps are actually suggested by IBM and it needs to take in practice of requirement gathering process In this research, 400 respondents responded for their view about useful traps for the betterment of requirement gathering process. 207 employees are strongly recommending for "Power up communication with visuals" because communications with visuals provides more visibility in requirement understanding. 104 employees are strongly agreed for "User of standard template to support requirement gathering work" as standard template is designed after considering best practices for requirement gathering process and hence it is quite useful for error free requirement gathering process. 88 employees strongly recommends "Avoid common pitfalls" means common mistakes needs to avoid while gathering collected needs from client. 86 employees are strongly responded "Use of Tools" option as using automated requirement gathering tools saves more time and lead to increase the productivity of requirement engineering team.

- **Involvement of different people in requirement gathering process:** There is need to involve many people like Senior management team, senior architecture team, testers,

developers, clients, end users and subscribers in the discussion of requirement gathering meeting or session. 100 percent of respondents are agreed for “end user” involvement and then for “Requirement Team”.

- **Time duration required to interact with end user for requirement gathering:** To estimate cost of any software project, time is bigger factor to consider in requirement gathering phase time required by designer or business analyst to interact with client becomes most important factor. If requirement is even though small but if it complex then business analyst may required more time to get it from end user. If client interaction time gets more as compare to estimated time then it may affect to delivery of software project or quality of project. 226 employees i.e 56.5 percent are strongly recommends daily 4 hrs required to interact with client or end user to understand requirement or to clarify requirement related queries.

- **Significance of Different types of Requirement Documents:** In requirement gathering session or meeting, business analyst needs to take collected needs verbally from client and then he/she needs to record or write requirement in specific document. Recording collected needs somewhere in document is a need of an hour because for further changes or future use we need base of requirement. It is seen that the highest average value is 7.71 for the ‘Functional Requirement Document (FRD)’ followed by ‘Component Design Document (CDD)’ that is 7.32 and ‘Component specification document (CSD)’ and ‘Test Case Document (TCD)’ which are 7.29 and 7.02. The average value for factor ‘Customer Requirement Document (CRD)’ is 6.78 followed by ‘Business Requirement Document (BRD)’ is 6.63. It is clear from the average values that Functional Requirement Document (FRD) is most important document as per most of respondents.

- **Time consumption of Business Analyst on non-requirement gathering activities:** In this research, following factors has been considered as non-requirement activities like Writing Requirement Documents, Reviewing FRD/BRD, Client Customer Interaction, Conducting Training for Testers and Developers are performed by business analyst and designer

To get views from respondents, data is divided in different ranges of percentage like 0-25, 25-50, 50-75, 75 and above. From the survey, it has been seen that higher percentage range value for 'Writing Requirement Document' is 129.

- **Factors responsible for Failure of Requirement Gathering process:** during requirement gathering process many challenges can encounter like changing nature of collected needs, inadequate communication, problem of scope, incomplete collected needs, ambiguous collected needs, wrong selection of stake holders, inappropriate selection techniques, conflicting collected needs are some of the problems It is seen that the highest average value is 7.52 for the 'Lack of knowledge about the business context' is followed by 'Lack of understanding of Business problems/opportunities' is 7.44, followed by 'Missing of gaps to be bridged' that is 7.18, followed by Inadequate Time' is 7.02 and 'Inadequate number of Resources' is 6.76.

- **Factors responsible to make software erroneous**: Software testing is a branch where verification of software's functionality is happening. Once development team complete their development and submit product to testing team, then testing team start verification of functionality of software via test cases execution. It is seen that the highest average value is 4.73 for the 'Requirement Errors' followed by 'Logic Design' is 4.34. The average value of 'Documentation' is 4.26 followed by 'Data' and 'Environment' that is 3.74. The average value of 'Interface' is 3.58 followed by 'Human' that is 3.16. So Requirement Errors' is most responsible factor to make software product erroneous because as we know that requirement is the base for all the phases of SDLC process and this information supports the ***Third Objective*** of the study.
- **Factors responsible for Failure of Software Project:** It is seen that the highest average value is 7.39 for the 'Lack of user involvement' followed by 'Poor or No Collected needs', 'Poor Testing' and 'Well-defined Schedules',' Long or unrealistic time scale', 'poor managerial decisions', 'Lack of foresight in building efficiency markets', 'Scope Creep' 'Cost overrun', 'Lack of methodology' ,'No Change Control System' ,'Inadequate Documentations' , 'Lack of an experienced'.

- **Awareness of Type of Software Testing**: In manual testing, tester needs to create test case, test data and manually execute test cases with dummy data on particular software component but in case of automated testing, test cases and dummy data has been created by testing tool itself. As per survey, total number of respondents who are doing Automated Testing is 239 and the total number of respondents who are doing manual testing is 161.

- **Defects raised by testing team per day:** If test cases are failed it means testers are raising defects against these failed test cases. Moreover, per day how many defects can be raised by testing team becomes important for management in terms of project completion. 100 percent testers are saying 5 to 6 defects are occurring per day.
- **Impact of poor requirement gathering on software testing process:** Poor requirement gathering is nothing but issues present in the collected collected needs from client or customer. A poor requirement is nothing but erroneous requirement .It is seen that the highest average value is 4.62 for the 'Addition of test cases' followed by 'Modification of test cases', 're-execution of modified test cases', 'Test result creation for newly added test cases', 'Deletion of test case', 'Verification of Newly added functionality due to Requirement Change'.

- **Overhead occurs in software testing due to poor requirement gathering:** Due to issues present in collected needs, many overheads can occur in software testing process. It is seen that the highest average value is 4.93 for the 'Gap in testing' is important overhead in testing process followed by 'inaccurate testing estimation' ,'System Testing Delay' , 'Test Team Credibility', 'Increase in system failures' , 'Delay benefit realization'.

- **Common Requirement Issues that may affect Software Testing:** Software is developed according to Clients Collected needs. Here some requirement issues are discussed with software developer and tester, which may affect software-testing process. It is seen that the highest average value is 4.73 for the 'Absence and Incompleteness' means if collected needs are incomplete then there is chance of having errors in software also, followed by 'Volatility', 'Incorrectness', 'Ambiguity and Vagueness' and 'Traceability'.

- **If collected SOP are not freezed, it has impact on software development process:** Software development effort estimation is the process of predicting the most realistic amount of effort (expressed in terms of person-hours or money or resource cost) required to develop or maintain software based on incomplete, uncertain and erroneous SOP from customer. Most of the employees are strongly agreed on Development and testing efforts must be carried out which requires extra cost during development of software.

- **Common SOP Issues that may affect Software business:** From each company 5 clients data is collected and analyzed their SOP collection duration. It is observed that for each company out of 5 clients atleast 3 clients are taking more time for SOP as their SOP is not freezed and eventually it has impact on software business.

B- FINDINGS

This research is related to relation or impact of requirement gathering on software testing process in software development process. The researcher has tested positively the hypothesis of this research study, with the help of primary and secondary data. The research findings are related to awareness and usage of testing and requirement gathering tools and finding hurdles in these processes.

- 400 respondent's data collected from different software company's resides in PMC and PCMC area. Among 400 respondent's, 47.5 percent employees belong to software companies reside in PMC area and 52.5 percent employees belong to software companies reside in PCMC area. **(Refer Chapter 4 Table 4.1. Software companies present in PMC and PCMC)**

- For the understanding and awareness about requirement gathering process, researcher has collected educational wise data from 400 employees of different software companies. For this research, this analysis has been done based on gender of employees. From the analysis,

it has been seen that 28.5 percent male employees and 5 percent female employees have designation as business analyst and 22.5 percent male and7.25 percent female employees having designation as Designer. Researcher also collected data for22.5 percent male and 14.25 percent females employees are working as a Tester. From this analysis, it has been proved that employees are working as Business Analyst, Designer, and tester in different software companies. **(Refer Chapter 4 Table 4.2: Gender and Occupation wise Distribution of Employees)**

- For this research, this analysis has been done based on qualification of employees. From the analysis, it has been seen that 32 percent postgraduate and 44.8 percent graduate employees are Business Analyst. 55 postgraduate and 96 graduate employees are Business Analyst. From this analysis, it has been proved that employees are graduate, postgraduate qualified, and they are having knowledge about requirement gathering process with designation wise experience.**(Refer Chapter 4 Table 4.3: Qualification and Occupation wise Distribution of Employees)**

- As we have considered PMC and PCMC areas from Pune city, Hadapsar, Kharadi and Shivaji nagar office locations belongs to PMC area and Hinjewadi belongs to PCMC area. Hence from the data analysis, it has been proved that 12.5percent employees are from Software companies which are located in Hadapsar area, 52.5 percent employees from Hinjewadi area, 9 percent from Kharadi and 26.25 percent from Shivaji Nagar. **(Refer Chapter 4 Table 4.4: Employees from different software companies present in different areas of pune city)**

- It is found that 92.25 percent of employees are agreed for 'Personal Meeting' requirement gathering technique to make error free software. **(Refer Chapter 4 Table 4.6 : Requirement Gathering Technique used in Software Industry)**
- It is found that the scope clarification for domain, input processes, reporting procedures, number of users and data collection are all the methods are most important and suitable for error free requirement collection. Using all these types of collected needs results into error free requirement specification, which in turn results into error free software project.**(Refer**

Chapter 4 Table 4.7 Types of collected needs need to use for betterment of software project)

- It is seen that more 51 percent employee are agreed for collecting requirement duration is 4 weeks because time duration is most important factor and playing vital role in the success or failure of any software project **(Refer Chapter 4 Table 4.8 Time Duration for Client interaction while gathering collected needs)**
- 52 percent of business analysts are agreed for use of "Power up communication with visuals" because communications with visuals provides more visibility in requirement understanding and to avoid this incorrect and incomplete requirement issue. **(Refer Chapter 4 Table 4.9 Useful traps for requirement gathering process)**
- 100 percent employees are agreed for involvement of End User and 98 percent employees agreed for involvement of Requirement Team in requirement gathering process for avoiding ambiguity in requirement. **(Refer Chapter 4 Table 4.10: People Involvement in requirement gathering process)**
- 57 percent employees insist that daily more than 4 hrs time should be used for interaction with client for gathering and understanding collected needs.**(Refer Chapter 4 Table 4.11: Time duration required to interact with end user for requirement gathering)**
- 83.5 percent developers agreed for use of Functional Requirement Document (FRD), this document is used to record all functional that is execution base detailed designs for software development process **(Refer Chapter 4 Table 4.12: Significance of Different types of Requirement Documents)**
- 70 percent developers insisting that 'Lack of knowledge about the business context' is most important factor which affects on requirement gathering process**. (Refer Chapter 4 Table 4.14: Factors affecting on Requirement Gathering)**
- 72 percent employees agreed that Requirement error is most important factor responsible for Software erroneous. **(Refer Chapter 4 Table 4.15: Factors responsible to make software erroneous)**

- It has found that 85 percent employees are agrees that 'Lack of user involvement' factor is responsible for failure of Software. **(Refer Chapter 4 Table 4.16: Factors responsible for failure of software project)**
- s94 percent Business Analyst agreed that 'IBM Rational Doors' tool is the best requirement gathering tools for requirement gathering, managing and analysis. **(Refer Chapter 4 Table 4.17 : Collected needs gathering Tools)**
- 59.75 percent testers agreed for use of testing tool during software testing process which is benefited for testing and minimizing errors.

  (Refer Chapter 4 Table 4.18: Employee's view about use of testing tools)
- Automated testing is good to get better productivity of testing team, hence there is need to know whether testers are using testing tools or not.57 percent are aware about to use of testing tools for testing of software. **(Refer Chapter 4 Table 4.19 : Employee's view about usage of testing tools)**
- 90 percent testers are agreed for execution of minimum 8 testcases per day for finding minute bugs present in the code to avoid further errors in software. **(Refer Chapter 4 Table 4.21 : Test cases execution per day)**
- 100 percent testers are agreed for finding minimum 5 to 6 defects from each testcases just to avoid further errors in software **(Refer Chapter 4 Table 4.22 : Defects raised per day)**
- 83 percent testers agreed that 'Test Case Document (TCD)' is a document useful for recording testcases which is useful for easily development of testcases.

  (Refer Chapter 4 Table 4.23: Significant Documents used in Software testing process)
- 84 percent Business Analyst agreed that cost required for software (tools)

  And resources like manpower, machine etc is important for to avoid failure of software project. Failures in software testing process always affect the quality of developed software and hence, software would become highest cost of software failures. **(Refer Chapter 4 Table 4.24: Cost Factors involved in testing process in terms of project failure)**
- 80 percent testers believed that Addition of TestCases during testing process is crucial task. Addition of testcases can be happened due to Poor requirement gathering.

  (Refer Chapter 4 Table 4.25: Work of Software testing process)

- More than 75 percent employees are agreed for if collected needs are not freeze, it has impact on software development process.

- Efforts required for task are considered as Development effort, Rework effort, Quality Assurance effort, Testing Effort. Software development effort estimation is the process of predicting the most realistic amount of effort (expressed in terms of person-hours or money or resource cost) required to develop or maintain software based on incomplete, uncertain and erroneous needs from customer. When needs from customers are volatile or keep on changing then Efforts of employees affected most because whichever task initially done by employees (earlier efforts) must be changed, so again employees are doing same work as per suggestions means development efforts i.e coding task , Rework effort i.e redesigning of product, Quality Assurance effort i.e product must be measured for its better quality, Testing efforts i.e whichever code has been changed by coder or programmer must be tested again by writing test cases , means all these efforts must be carried out for changes nothing but it has impact on cost which can be counted as "Impact on Business". **(Refer Chapter 4 Table 4.26: Efforts carried out in case collected needs are not freeze)**
- Around average values 4.93 of respondents are agreeing that poor collected needs always create in gap in testing work of testers. Gap in testing or discontinuity in tester work lead to wastage of time of testing resources and it lead to Inaccurate testing estimation. **(Refer Chapter 4 Table 4.27: Overhead occurs in software testing due to poor requirement gathering)**
- 100 percent software employees are agreed that 'Absence and Incompleteness' factor affect software testing. Hence, researcher of this research recommends that requirement should be complete or should not be missing any important part. **(Refer Chapter 4 Table 4.28: Common Requirement Issues that may affect Software Testing)**
- From each company 5 clients data is collected and analyzed their SOP collection duration.

  researcher has observed that for each company out of 5 clients atleast 3 clients are taking

  more time for SOP as their SOP is not freezed and eventually it has impact on software

business. **(Refer Chapter 4 Table 4.29: Software Development Life Cycle during Year 2015-2016)**

Chapter 6. Conclusion and Suggestions

1. Personally meeting with clients, through documents, online or automated are equally important techniques for collecting unambiguous needs from customer in software development process.
2. Type of customer's needs i.e Using Scope Clarification for Domain, Input Processes, Reporting Procedures, Number of Users and Data Collection are important information which results into error free software product.
3. Time for collecting customer needs required for product development must be Adequate so that proper needs can be collected to avoid further problems on product.
4. "Power up communication with visuals" is useful and effective technique for collecting customer needs through this technique proper set of needs can be gathered.
5. Many people like senior management team, senior architecture team, testers, developers, clients, end users and subscribers are involved in the discussion of collecting needs.
6. Time is the most important factor for software development process because as time increases, schedule may get lagged due to which it may have indirect effect on cost and business.
7. 'Lack of knowledge about the business context' is the most responsible factor for failure of collecting needs from customer process.
8. 'Requirement Errors' is the most responsible factor to make software product erroneous because as we know that requirement is the base for all the phases of SDLC process.

9. ‘Lack of user involvement’ followed by ‘Poor or No Requirements’ is most important factor responsible for the failure of software project.
10. ‘IBM Rational Doors’ and ReqHarbor.com are the best collecting needs tool through which needs are stored in automated format and can be accessed by team of software product development.
11. Automated Testing is the most useful method of software testing in software companies.
12. Execution of 8 testcases per day using automated testing mode is best practice for software testing process to avoid failure of software product.
13. Tester should expect 5 to 6 defects per day in their software testing process to avoid software failure.
14. “Addition of test cases” is the most frequent task tester needs to do when customer’s needs are incomplete and changing though its development process.
15. ‘Gap in testing or discontinuity in testing work’ affects total cost of software project. Cost is indirectly related with schedule of development.
16. Absence and Incompleteness, Incorrectness, Ambiguity and Vagueness, Volatility, Traceability parameters are the most important causes for failure of software.
17. Maximum companies who spent more time on collecting customer’s needs incur more cost as cost is indirectly related to time for product development.

18. **Thus the final conclusion is noticed through the research. The process of collecting accurate needs should be well documented to resolve ambiguity because it directly impacts on business of software development. The process of collecting needs must**

be automated through tools so that ambiguity gets resolved and proper development process will get executed. Once needs gets freezed there should not be delay in schedule for development and thus extra cost should not be incurred for whole software development process.

Suggested Model

Considering the present state of impact of collection of SOP process need to design through the present research work. This new model is designed named as called Customer's Needs Management to Reduce Software Failures Model (CNMRSF)

The main functionality of CNMRSF model is to provide better software testing actions for corresponding poor requirement. Model has 3 phases like input, processing and output. CNMRSF model integrates the functionality of different modules like Input module, processing module and output module

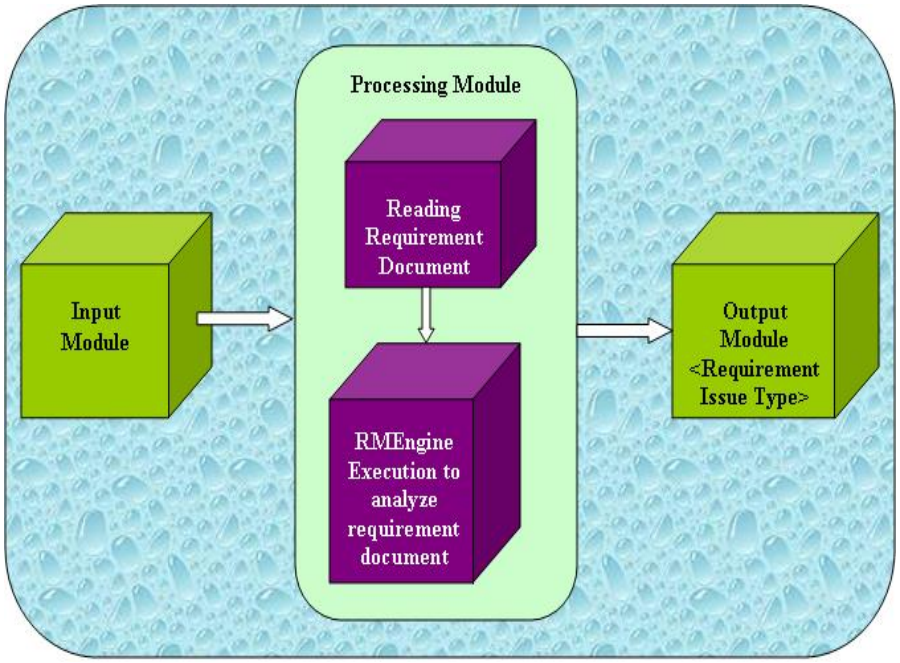


Fig 1. Suggested Model for Requirement gathering and analysis process.

Input Phase

Input phase will contain input module which deals with input data collection from the end user and this input data will be in the form of functional requirement document (FRD) or Customer Requirement Document (CRD). In this phase, Requirement document can get from your local computer drive. Main functionality of this phase is to get exact type and requirement document and call the processing module for further analysis.

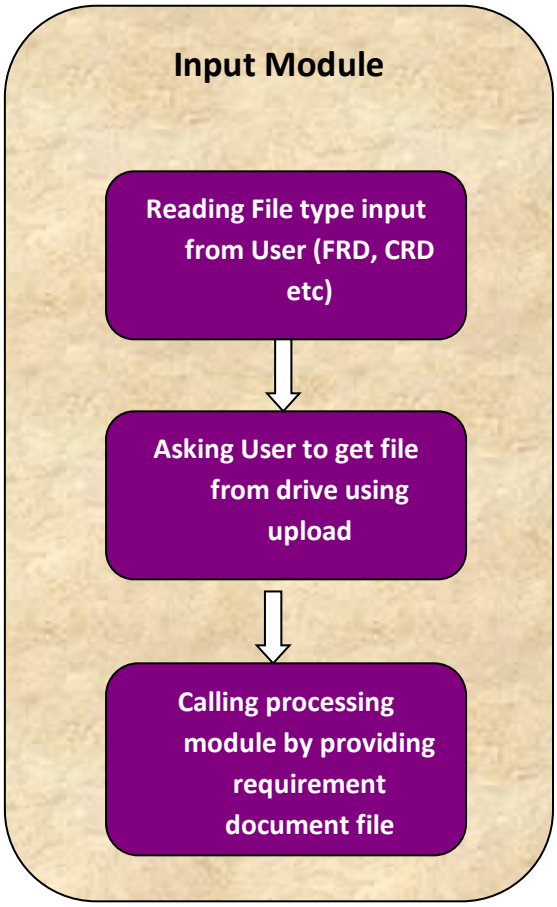


Fig 2. Input module of CNMRSF Model Processing Phase

Processing phase deals with integrated functionality of Reading Requirement document, analyzing requirement document by call Requirement Management Engine

(RAEngine) and Execution of Output module to generate list of Issues, Impacts and Actions for particular type of requirement issue. Processing phase contains processing module which will deal with integrated functionality of Reading Requirement document, analyzing requirement document by call Requirement Management Engine (RAEngine) and Execution of Output module to generate list of Issues, Impacts and Actions for particular type of requirement issue.

In processing phase, RMEngine gets call to analyze exact requirment issue type. Requirement Analysis Engine (RAEngine) will be the major part of processing module and will be developed based on the requirement issues responded by respondent from different software companies. For RMEngine, following requirement issues will be considered [11].

1. Incomplete/Absent
2. Incorrect
3. Ambiguity & Vagueness
4. Volatility
5. Traceability

RAEngine is heart of CNMRSF module. Without RAEngine, CNMRSF can not do anything. RAEngine is basically works on if else ladder concept. It first checks what is exact requirment issue present in provided requirment document and based on that decide type of requirement issue. For deciding appropriate requirement issue, RAEngine analyze requirment document by compairing it with software system architechure document and tries to provide exact requirement issue. RAEngine takes Requirment document as an input and generates requirement issue type by considering many isssues present in provided requirement document.

As metioned above, RAEngine maninly focuses on five type of requirment issues like incomplete,incorrect, Ambuiguity,vaguess, volatility and traceability etc. [4]. Based on different conditions like if track table is missing or improper change control process found then its mark requirement issue as traceability issue. If functional or non-functional collected needs are missing then it marks requirment issue type as incorrect. If there is change between old requirment document and new requirement document then it marks requriement issue as

volatility. If requirement followed poor requirement definitoin then it marks requirement issue as Ambiguity and Vagueness. Like wise it checks for Incomplete/Absence requirement issue. Here using if syntax RAEngine verifies many conditions to decides appropriate requirment issue.

Once requirment issue is identified by RAEngine, it returns that requirement issue back to processing module and then processing module works on further analysis.

Output generation phase

Output phase deals with generation of output based on input argument as a requirement issue type provided by processing phase. Output phase has module named as Output Module and this module is basically gets executed by Processing module. Main functionality of output module is to get requirement type issue as an input and based on this input query to database to fetch corresponding list of Issues, Impacts and Action points. This module is displaying list of Issues, Impacts and Actions based on corresponding requirement issue type.

Output module deals with database to fetch records from three different tables named as Issue, Impact and Action. The perquisite of this model is these tables should get created with data in database.

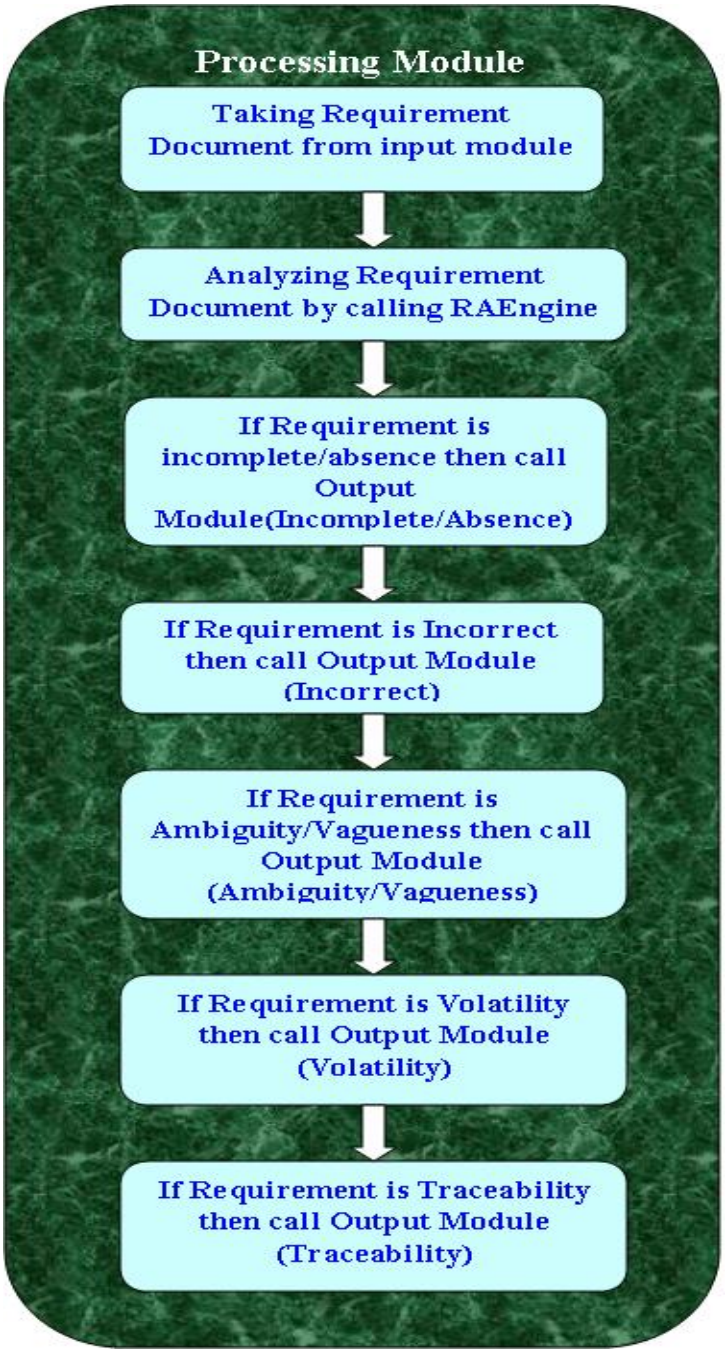


Fig.3 Processing Module

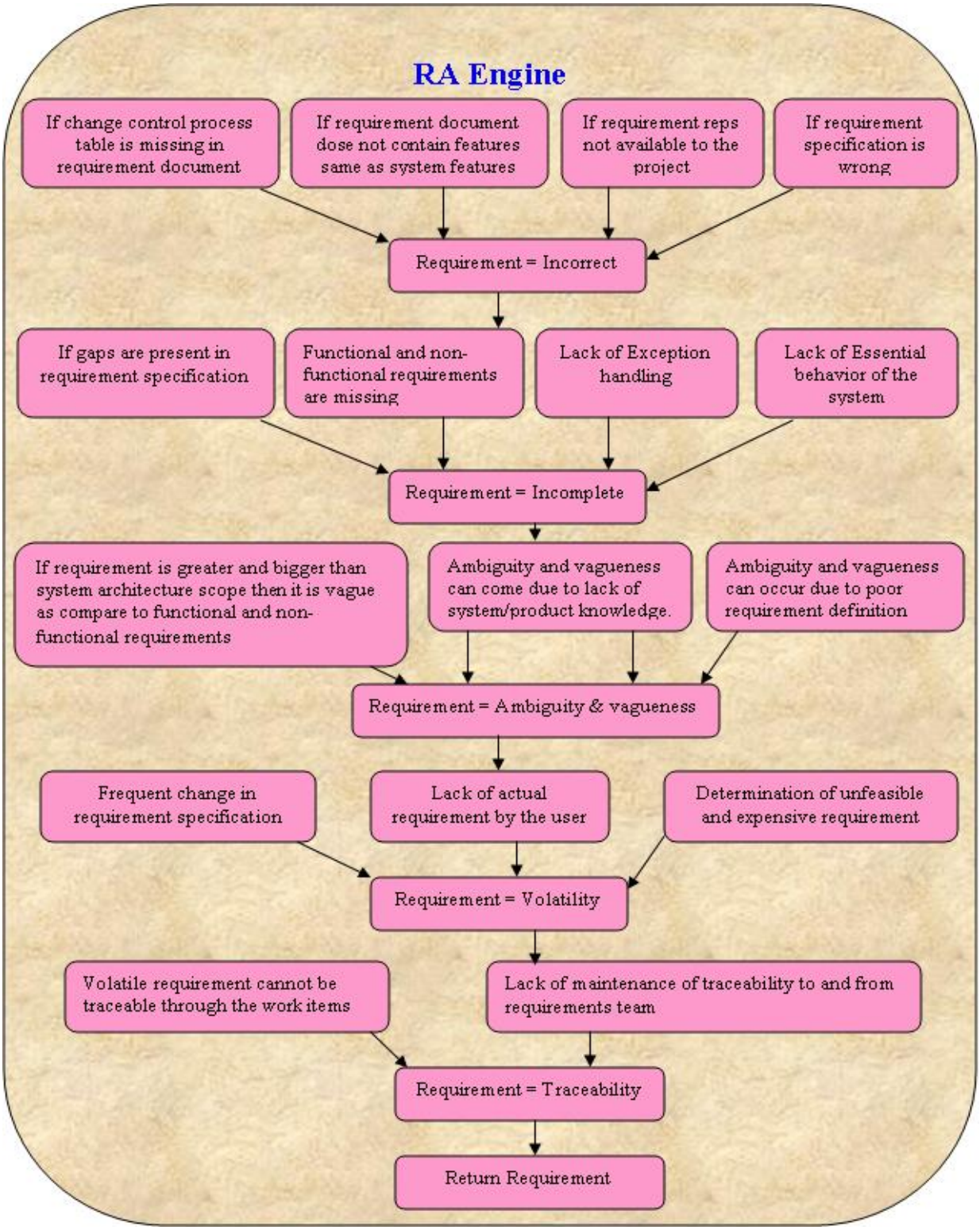


Fig 4. Requirement Engine

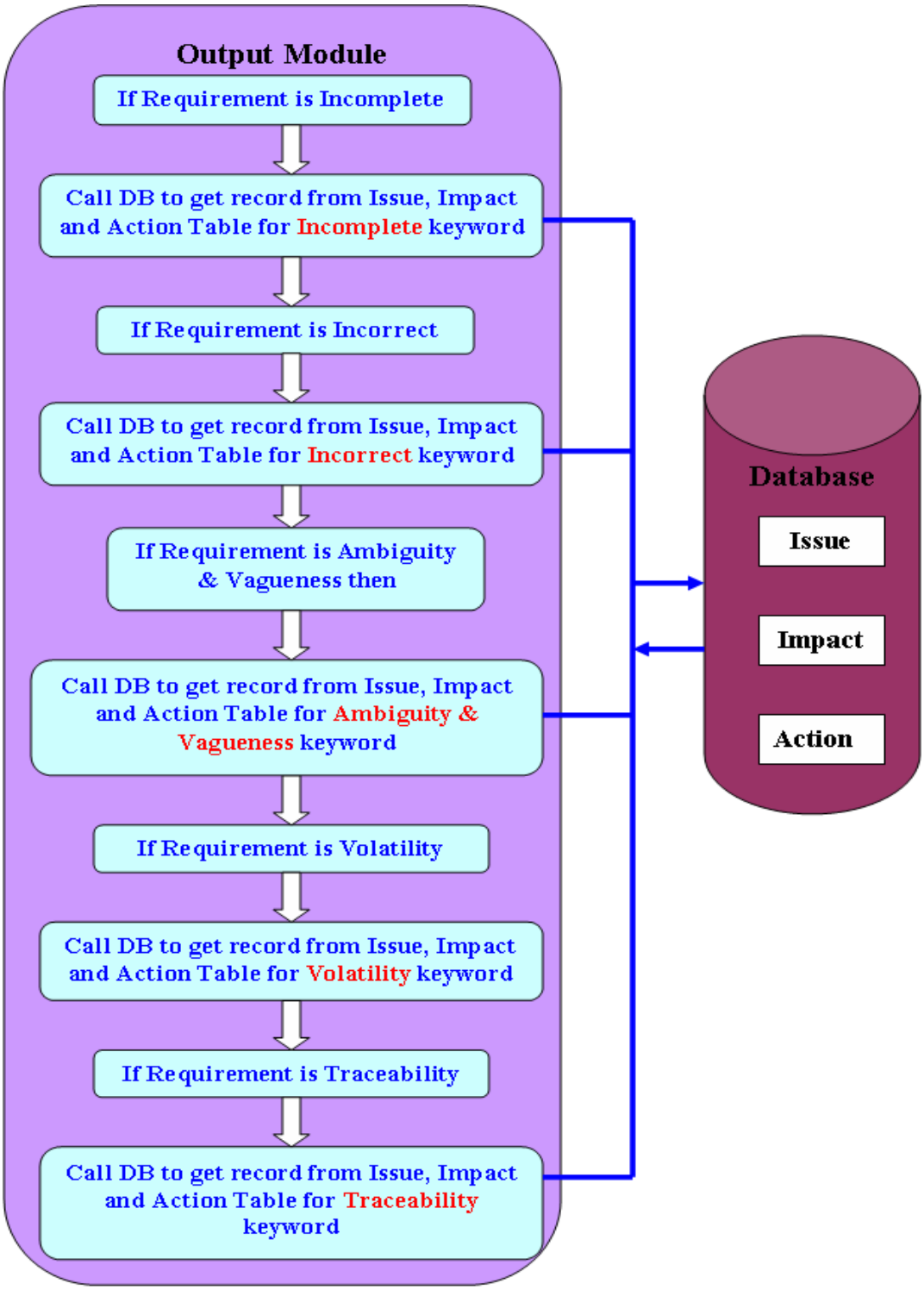


Fig 5.Output Module

SUGGESTIONS

1. It is suggested that developers, software testers should read and understand customer's needs carefully before starting their work.
2. Management of any software company should have adequate number of resources, work allocation between resources should be balanced and requirement engineer should also not spend more time collecting needs from customer.
3. Collected needs must be accurate and well documented.
4. Customer needs must be collected through automated tools.
5. Researcher has suggested a format of documents for collecting customer needs in the proposed model.
6. Software testers should write testcases for accurate customer's needs.
7. Software tester should get involved in the requirement phase and also communicate with requirement engineers for better understanding of customer requirements.
8. Software tester should create correct test cases and test data before starting software testing.
9. Software tester should verify test results with exact functionality required by customers and also consider performance of software system.
10. Software companies should consider all the factors which are responsible for failure and rectify the same immediately.
11. Implementation of model improves the interaction between developer and tester and helps to increase quality of the product.

12. Encourage the software companies for active participation in quality product development and implementation of model in minimal cost.
13. Interaction of researcher from Industry and academia is also required to make constant improvement for successful implementation of model for better quality of product.
14. Conduction of quality audit from third party
15. Software Testing Clubs participation in execution of quality audits with standardized (ISO, CMM, Six Sigma etc) companies.
16. Awareness about the quality standards among the employees of the software companies can be created.
17. To make the employees more productive, thrust on awareness about tools by arranging various training sessions for employees.
18. Make employees aware of their responsibility towards development of quality product.
19. There should be QA team activity on feedback system for employees on quality development, tester performance improvement.
20. Organize quality product fest program to create awareness about quality product among the employees.
21. QA team should organize award and recognition fest for successfully and error free development of software.
22. Active involvement of finance manager throughout SDLC will help in keeping track of the cost.
23. The manuals must be provided and followed by employees during implementation of model to avoid errors.

24. There should be up gradation of tools used for development and testing.
25. Organization should purchase upgraded version and licensed of various tools used in software companies.
26. By using tools organization saves resources like time, efforts, and cost.
27. By using automated reverse engineering tool requirement changes can be easily traced out in the development process.
28. In suggested model, input model stores all customer's needs document category wise like missing requirement, wrong requirement etc. which will help to avoid errors in the product.
29. Timely freezing of customer's needs will lead to better utilization of resources like cost and time.

SCOPE OF FUTURE

- More requirement documents formats will be supported by Customer Needs Management to Reduce Software Failures model.
- More detail analysis of requirement documents and recognition of best requirement issue by using by Customer Needs Management to Reduce Software Failures model.
- More factors will be considered for the failure or success of software projects.
- More clarity will be focused on to reduce the failure of software project.
- More actions will be provided by Requirement Management to Reduce Software Failures model for software requirement engineers, developers, testers etc.

Bibliography:

1. The Standish Group Report (CHAOS). (2003). Retrieved November 2013, from: http://www.projectsmart.co.uk/docs/chaos-report.pdf.

2. Boehm, B., & Bose, P. (1994). A collaborative spiral software process model based on Theory W. Third International Conference on the Software Process, pp. (59-68).
3. Cao, L., & Ramesh, B. (2008). Agile Collected needs Engineering Practices: An Empirical Study. IEEE Software, 25 (1), pp. (60-67).
4. 2013, Md Rounok Salehin "Missing Collected needs Information and its Impact on Software Architectures:A Case Study" The School of Graduate and Postdoctoral Studies The University of Western Ontario,London, Ontario, Canada
5. Gross, A., Doerr, J. (2012). What do software architects expect from collected needs specifications? results of initial explorative studies. IEEE First International Workshop on Twin Peaks of Collected needs and Architecture, IEEE Software, pp. (41-45).
6. Lee, S., & Rine, D. (2004). Missing Collected needs and Relationship Discovery through Proxy Viewpoints Model. 19th annual ACM Symposium on Applied Computing, pp. (1513-1518). Nicosia, Cyprus.
7. George, B., Bohner, S. A., & Prieto-Diaz, R. (2004). Software information leaks: A complexity perspective. Ninth IEEE International Conference on Engineering Complex Computer Systems (ICECCS'04), pp. (239-248). Florence, Italy: IEEE Computer Society.
8. Gumuskaya, H. (2005). Core Issues Affecting Software Architecture in Enterprise Projects. Proceedings of World Academy of Science, Engineering And Technology, volume 9, pp. (35-41)
9. Thomas Kiihne , "What is Model?" Darmstadt University of Technology, Darmstadt, Germany.
10. http://www.la1.psu.edu/cas/jgastil/pdfs/conceptualdefinitiondeliberation.pdf
11. Kirsten Kiefer, "The Impact of Requirement issues on testing", Software Education associates Ltd
12. Brooks, F. 1987. No Silver Bullet: Essence and Accidents of Software Engineering. IEEE Computer, Vol. 20, No. 4, April 1987, 10-19.
13. Jayaswal, B. K., Patton, P. C. 2006. Design for Trustworthy Software: Tools, Techniques, and Methodology of Developing Robust Software, 1st Edition. (September 2006), Prentice Hall edition.

14. Zowghi, D. 2002. A Study on the Impact of Collected needs Volatility on Software Project Performance. Proceedings of Ninth Asia-Pacific SE Conference (APSEC‟ 2002), IEEE Computer Science.
15. Taghi, M., Khoshgoftaar, N., Sundaresh, N. 2006. An empirical study of predicting software faults with case-based reasoning. (June 2006), Software Quality Control.
16. Jiang, Y., Cukic, B., Ma, Y. 2008. Techniques for evaluating fault prediction models. (October 2008), Empirical Software Engineering.
17. M.P.Singh, Rajnish Vyas, "Collected needs Volatility in Software Development Process" International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-2, Issue-4, September 2012
18. Zowghi, N. Nurmuliani, ―A study of the Impact of collected needs volatility on Software Project Performance, Proceedings of the Ninth Asia-Pacific Software Engineering Conference , APSEC 2002, Gold
19. Cost, Queensland, Australia,04-06 Dec 2002, pp:3-11.
20. http://www.mapsofindia.com/maps/maharashtra/pune.htm (23/7/2008)
21. http://www.mapsofindia.com/pune/software-company-pune.html
22. https://maps.google.co.in/maps?hl=en-IN&gbv=2&ie=UTF-8&fb=1&gl=in&q=software+companies+in+pune&hq=software+companies&hnear=0x3bc2bf2e67461101:0x828d43bf9d9ee343,Pune,+Maharashtra&ei=av_nVKH6O86IuwSvrIL4Bw&ved=0CB4QtQM&output=classic&dg=brw
23. Sr. S. P. Gupta, "Statistical Methods", Sultanchand & Sons Publication, New Delhi.
24. Don Gotterbarn, "Reducing Software Failures: Addressing the Ethical Risks of the Software Development Lifecycle" Australian Journal of Information Systems
25. Don Gotterbarn, "Reducing Software Failures: Addressing the Ethical Risks of the Software Development Lifecycle" Australian Journal of Information Systems
26. Muhammad Naeem Ahmed Khan and et.all (2013), "Review of Collected needs Management Issues in Software Development" I.J.Modern Education and Computer Science, 2013, 1, 21-27, Published Online January 2013 in MECS (http://www.mecs-press.org/) DOI: 10.5815/ijmecs.2013.01.03

27. https://web.cs.dal.ca/~hawkey/3130/SEBackground4.pdf
28. Systems Development Lifecycle: Objectives and Collected needs. Bender RPT Inc. 2003
29. Vanshika Rastogi (2015), “Software Development Life Cycle Models- Comparison, Consequences” IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 6 (1) , 2015, 168-172
30. Software Development Life Cycle (SDLC) Yogi Berra presentation
31. Ms. Shikha maheshwari and Prof.Dinesh Ch. Jain 2012 “A Comparative Analysis of Different types of Models in Software Development Life Cycle” International Journal of Advanced Research in Computer Science and Software Engineering, Volume 2, Issue 5, May 2012
32. Royce, Winston (1970), "Managing the Development of Large Software Systems" (PDF), Proceedings of IEEE WESCON 26 (August): 1–9
33. PK.Ragunath, S.Velmourougan, P. Davachelvan, ,S.Kayalvizhi, R.Ravimohan (2010) “Evolving A New Model (SDLC Model-2010) For Software Development Life Cycle (SDLC)” IJCSNS International Journal of Computer Science and Network Security, VOL.10 No.1, January 2010
34. Seema, Sona Malhotra 2012 “Analysis and tabular comparision of popular SDLC models” International Journal of Advances in computing and Information Technology.
35. Sonali MAthur and Shaily Malik (2010), “Advancements in the V-Models”, International Journal of Computer Applications (0975-8887) Volume 1- No.12
36. Naresh Kumar, A. S. Zadgaonkar, Abhinav Shukla “Evolving a New Software Development Life Cycle Model SDLC-2013 with Client Satisfaction” International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-3, Issue-1, March 2013
37. G. Kotonya and I. Sommerville, (1998), have published article on “Collected needs Engineering: Processes and Techniques”, in the book published by Chichester, UK: John Wiley & Sons.
38. Collected needs Engineering A good practice guide, Ramos Rowel and Kurts Alfeche, John Wiley and Sons, 1997

39. I. Sommerville and P. Sawyer (1997), Collected needs Engineering: A Good Practice Guide, New York: John Wiley & Sons,.
40. http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=948567&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D948567
41. K. E. Wiegers, Software Collected needs, 2nd ed., Redmond, W A: Microsoft Press, 2003.
42. http://prr.hec.gov.pk/Chapters/369S-2.pdf (Software Testing reference)
43. Donald Firesmith, Software Engineering Institute, U.S.A "Common Collected needs Problems, Their Negative Consequences, and the Industry Best Practices to Help Solve Them". http://www.jot.fm/issues/issue_2007_01/column2/
44. Olga Liskin , et al., "Supporting Acceptance Testing in Distributed Software Projects with Integrated Feedback Systems: Experiences and Collected needs" 2012 IEEE Seventh International Conference on Global Software Engineering
45. Vishawjyoti, Sachin Sharma, "Study and Analysis of automation testing techniques" , Journal of global research in computer science, Volume 3, No. 12, December 2012, ISSN-2229-371
46. Antonia Bertolino has published his research article on "Software testing research and practice"
47. Vivek Kumar (2012) has published his article on "Comparison of Manual and automation testing" International Journal of Research in Science And Technology, (IJRST) 2012, Vol. No. 1, Issue No. V, Apr-Jun, ISSN: 2249-0604
48. R. M. Sharma (2014), "Quantitative Analysis of Automation and Manual Testing" International Journal of Engineering and Innovative Technology (IJEIT) Volume 4, Issue 1, July 2014
49. Harsh Bhasin, at.el. (2014), have published research article on "Black Box Testing based on Requirement Analysis and Design Specifications"
50. MIRZA MAHMOOD BAIG (2009), "NEW SOFTWARE TESTING STRATEGY" N.E.D. University of Engineering & Technology

51. Mohd. Ehmer Khan , Farmeena Khan “A Comparative Study of White Box, Black Box and Grey Box Testing Techniques”, (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 3, No.6, 2012
52. Paul C. Jorgensen (2013) published book on “Software testing: a craftsman's approach” CRC Press
53. Wasif Afzal et al. (2008) “A Systematic Mapping Study on Non-Functional Search-based Software Testing” paper available at http://www.researchgate.net/publication/221391274
54. W. K. Chan et al (2002), have published article on “An Overview of Integration Testing Techniques for Object-Oriented Programs” Proceedings of the 2nd ACIS Annual International Conference on Computer and Information Science (ICIS 2002), International Association for Computer and Information Science, Mt. Pleasant, Michigan (2002)
55. Shivkumar Hasmukhrai Trivedi, (2012), has published research article on “Software Testing Techniques” International Journal of Advanced Research in Computer Science and Software Engineering, Volume 2, Issue 10, October 2012
56. Leung, H.K.N (1989) has published article on “Insights into regression testing” Software Maintenance, 1989., Proceedings., Conference on
57. M. Fagan, “Design and Code Inspections to Reduce Errors in Program Development,” IBM Systems Journal, vol. 38, no. 2/3, pp. 258–287, 1999.
58. Hitesh Tahbildar at el(2011). “Automated software test data generation: Direction of research” International Journal of Computer Science & Engineering Survey (IJCSES) Vol.2, No.1, Feb 2011
59. http://www.rishabhsoft.com/blog/beta-testing-the-importance (2011)
60. Ms. S. Sharmila, “Analysis of Performance Testing on Web Applications” International Journal of Advanced Research in Computer and Communication Engineering Vol. 3, Issue 3, March 2014

61. Pooja Ahlawat (2013) “A Comparative Analysis of Load Testing Tools Using Optimal Response Rate” International Journal of Advanced Research in Computer Science and Software Engineering. Volume 3, Issue 5, May 2013

62. Chapter 4 ‘Capturing the Collected needs’, http://www.cse.msu.edu/~chengb/RE-491/Papers/atlee-chapter4.pdf

63. 2013, Md Rounok Salehin “Missing Collected needs Information and its Impact on Software Architectures:A Case Study” The School of Graduate and Postdoctoral Studies The University of Western Ontario,London, Ontario, Canada

64. Mohd. Ehmer Khan, “Different Forms of Software Testing Techniques for Finding Errors,” IJCSI, Vol. 7, Issue 3, No 1, pp 11-16, May 2010

65. Christel, Michael and Kyo C. Kang (September 1992). "Issues in Collected needs Elicitation". Technical Report CMU/SEI-92-TR-012. CMU / SEI. Retrieved January 14, 2012.

66. Donald Firesmith, Software Engineering Institute, U.S.A “Common Collected needs Problems, Their Negative Consequences, and the Industry Best Practices to Help Solve Them”. http://www.jot.fm/issues/issue_2007_01/column2/

67. Indika Perera, “Impact of Poor Requirement Engineering in Software Outsourcing: A Study on Software Developers’ Experience”. Int. J. of Computers, Communications & Control, ISSN 1841-9836, E-ISSN 1841-9844 Vol. VI (2011), No. 2 (June), pp. 337-348

68. Collected needs Engineering A good practice guide, Ramos Rowel and Kurts Alfeche, John Wiley and Sons, 1997

69. Chapter 4 ‘Capturing the Collected needs’, http://www.cse.msu.edu/~chengb/RE-491/Papers/atlee-chapter4.pdf

70. http://www.tutorialspoint.com/software_engineering/software_engineering_overview.html

71. http://www.jot.fm/issues/issue_2007_01/column2.pdf

72. http://www.enterprisecioforum.com/en/blogs/pearl/it-project-failure-symptoms-and-reasons.

73. IndikaPerera, “Impact of Poor Requirement Engineering in Software Outsourcing: A Study on Software Developers’ Experience”. Int. J. of Computers, Communications & Control, ISSN 1841-9836, E-ISSN 1841-9844 Vol. VI (2011), No. 2 (June), pp. 337-348

74. Collected needs Engineering A good practice guide, Ramos Rowel and KurtsAlfeche, John Wiley and Sons, 1997

75. **Donald Firesmith,** Software Engineering Institute, U.S.A **“**Common Collected needs Problems, Their Negative Consequences, and the Industry Best Practices to Help Solve Them”. http://www.jot.fm/issues/issue_2007_01/column2/

76. Christel, Michael and Kyo C. Kang (September 1992). "Issues in Collected needs Elicitation". Technical Report CMU/SEI-92-TR-012. CMU / SEI. Retrieved January 14, 2012.

77. https://www.utdallas.edu/~chung/RE/Getting_collected needs_right-avoiding_the_top_10_traps.pdf

78. Vidya Gaveakr, (2013) “A Study of Geographic Information System based computerized framework to enhance the water supply system in Pune City” Thesis of Computer Management Dept., Tilak Maharashtra Univerisity.

79. http://www.enterprisecioforum.com/en/blogs/pearl/it-project-failure-symptoms-and-reasons.

80. K.K. Aggarwal and Yogesh Singh,”Software Engineering”,New age International Publishers, third Edition, 2008.

81. http://www.umsl.edu/~sauterv/analysis/Fall2010Papers/Isserman/

82. http://www.practicalecommerce.com

83. http://www.reqharbor.com/

84. https://www.google.co.in/webhp?hl=en#hl=en&q=data%20dictionary

85. https://ankitmathur111.wordpress.com/2012/06/20/whats-whys-hows-of-software-testing-wwh/
86. http://istqbexamcertification.com/what-is-fundamental-test-process-in-software-testing/
87. Thomas Kiihne , "What is Model?" Darmstadt University of Technology, Darmstadt, Germany.
88. http://www.la1.psu.edu/cas/jgastil/pdfs/conceptualdefinitiondeliberation.pdf
89. http://www.voltreach.com/Development_Methodologies.aspx
90. Kirsten Kiefer, "The Impact of Requirement issues on testing", Software Education associates Ltd.
91. http://www.softed.com/resources/docs/impact-of-collected needs-issues-on-testing.pdf
92. http://blog.sei.cmu.edu/post.cfm/common-testing-problems-pitfalls-to-prevent-and-mitigate
93. http://kinzz.com/resources/articles/91-project-failures-rise-study-shows
94. S. Arun Kumar and T.Arun Kuma "Study The Impact Of Collected needs Management Characteristics In Global Software Development Projects: An Ontology Based Approach" in International Journal of Software Engineering & Applications (IJSEA), Vol.2, No.4, October 2011
95. http://www.pune.ws/in/?list=it_software_companies-hinjawadi|here
96. http://www.pune.ws/in/?list=hadapsar-it_software_companies
97. http://www.pune.ws/in/?list=it_software_companies-kharadi
98. http://www.pune.ws/in/?list=it_software_companies-magarpatta
99. http://hiapune.in
100. Karl E. Wiegers More About Software Collected needs: Thorny Issues and Practical Advice(Microsoft Press, 2006; ISBN 0-7356-2267-1)Chapter 2: Truths About Software Requirement

101. Henry Johnson, An approach to software project management through collected needs engineering, At Texas Tech University, Henry Johnson, December 2010

102. Davis , C.J, Fuller, R.M. Tremblay, M.C. & Berndt, D.J. (2006). Communication Challenges in collected needs elicitation and the use of the repertory grid technique. Journal of computer information Systems, 78

103. Bourque, P.; Fairley, R.E. (2014). "Guide to the Software Engineering Body of Knowledge (SWEBOK)". IEEE Computer Society. Retrieved 17 July 2014

104. https://en.wikipedia.org/wiki/Software_development_effort_estimation

105. "Impact of software requirement volatility pattern on project dynamics: evidences from a case study" International Journal of Software Engineering & Applications (IJSEA), Vol.2, No.3, July 2011

106. https://www.dnb.co.in/TopIT/company_listing.asp?PageNo=1&q=employee&r

Books

1. Software Engineering for Students- A Programming Approach by Douglas Bell Pearson Education. Pg 230-255
2. Software Engineering – A Practitioners Approach by Roger S. Pressman Tata McGraw Hill.
3. Quality, 5th ed., Prentice-Hall, 2010. Donna C. S. Summers. Pg 20-57
4. Total Quality Management, Prentice Hall, 2003 Dale H. Besterfield. . Pg 37-77
5. Information Technology Project Management -Kathy Schwalbe. Pg 7-177
6. Software Metrics A rigorous and practical approach – N Fenton, S Lawrence Pfleeger. Pg 170-255
7. .Research Methodology Methods and Techniques By C R Kothari and Gaurav Garg. Pg 52-109
8. .A Practitioner's Guide to Software Test Design, Lee Copeland, 2003. Pg 38-97
9. The Art of Software Testing, 2nd edition, Glenford Myers, et. 2004. Pg 76- 143

10. Software Testing Techniques, 2nd edition, Boris Beizer, 1990. Pg 14-65

11. How to Break Software: A Practical Guide to Testing, James Whittaker, 2002.

Pg123-254

Research Student **Research Guide**

Mrs.Ashvini Shende **Dr.Prasanna Deshmukh**

Chapter 1

Introduction

1.1. Introduction

This chapter gives explanation about basic terms used in thesis for the topic "A Study of Collecting customer's SOP in software development process and its impact on Business of selected IT Companies". [Annexure-II]

Customer SOP are also referred as ***Statement of Purpose*** for software development Process.

21st century is known for computerization of all manual works, that human being was doing so far. Computerization made man's life easy and this computerization became possible because of integration of hardware and software. Software plays most important role in the automation of most of electronic appliances. Hence, in current market, demand for all types of software is increasing day by day. This demand leads to development of thousands of software applications that in turn increases number of software industries in overall world.

Software project is nothing but collection of larger programmers with many interactions and functional dependencies. It involves creation of a product that has never been created before although the development processes are similar among other projects. As a result, software development projects have a dismal track-record of cost and schedule overruns and quality and usability problems [3]. In software industry, software project terminology is used for the complete life cycle of these software applications. Each customer's SOP belongs to each software project.

Software development projects are frequently time-driven, and project managers are often put in the undesirable position of having to reduce the time to market. Combined with the common misconception that a software project does not truly start until coding begins, this notion leads to shortening, or even eliminating the preliminary analysis and planning for the project. Presumably, software project managers do this because they believe this practice will increase the success of the software project, with only a minimal effect on the product's quality.

Software Development Life Cycle [27]

Software Development Life Cycle, also known as SDLC holds a very strong position in the software development process. The SDLC helps one to determine or come up with an approximate time that will be required to develop a software. Also, it helps in determining the various phases through which a software undergoes throughout its life or while it is being developed. Software Development Life Cycle has various stages or phases, each of which has a specific task and definition. Each process is a successor of the previous one, i.e., the phases should be executed in a specific manner.

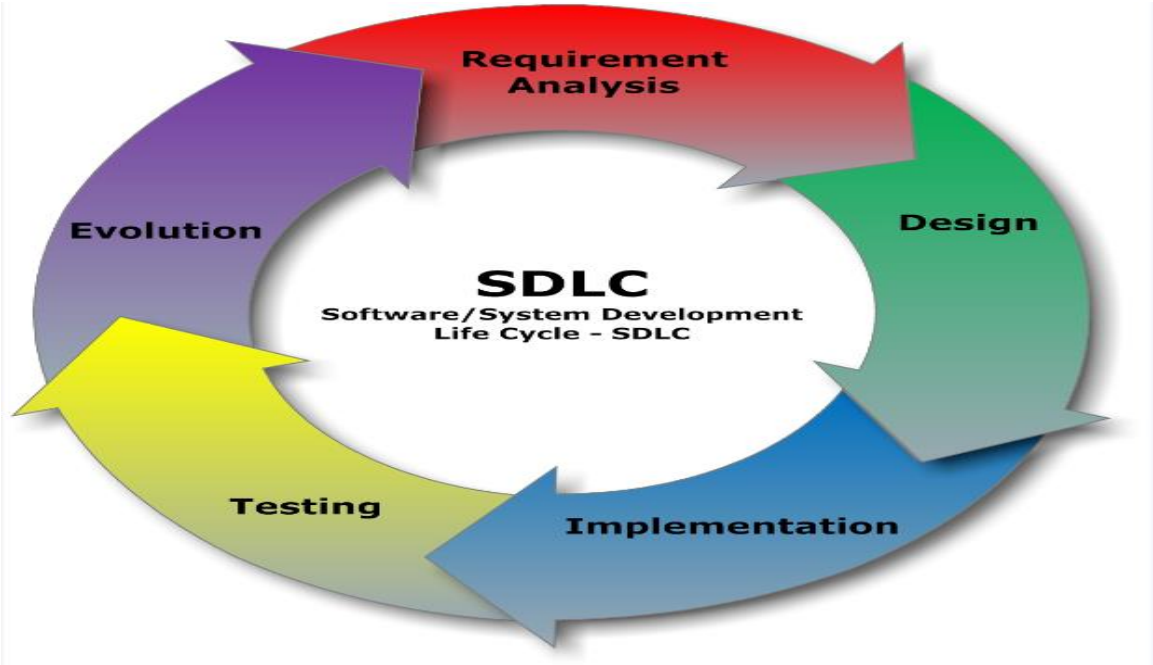


Fig. 1.1 Software Development Life Cycle

Following are the different stages of SDLC:

1. Requirement gathering & Analysis: The requirement from the customer are gathered. Various questions like who is going to be the user of the product? What does the software do? What type of data is to be stored? Etc. helps in gathering the information from the customers. Once the customer's SOP is gathered properly, it is then analyzed and feasibility is checked to make sure if it a good idea to move forward with.
2. Design: In this phase, the overall design of the product, i.e., the software and the system is prepared based on the study of the customer's SOP gathered in the previous phase.
3. Implementation/ Coding: The software is then coded according to the design made.
4. Testing: The developed product is then tested for the desired result and output. Various test cases are applied on the software so developed to check if it gives the desired output in every test case.
5. Deployment: Once the product undergoes the testing phase, it is deployed, i.e., delivered to the customer along with all the necessary documents such as user manual. This phase also consists of beta testing part. Beta testing is done by the users of the product.
6. Maintenance: Once the software is deployed to the customer, the tech team along with the after sales team takes care of the software and any complaints from the user of the software.

The key difference between engineering a software product and a hardware product is the greater degree to which the intangible product must be conceptualized before it is built. The first step to accomplishing this is clearly defining and understanding the user's problem. Unlike a bridge or a building, software is an idea, and is therefore relatively free from physical constraints of the real world. Computer logic offers many more possibilities for perfect solutions, along with many more possibilities for bad ones. While avoiding bad solutions is important in the physical world as well, it is much harder to

accomplish when the number of possible solutions is greater. One of the key ways to avoid choosing a bad solution is by clearly defining and understanding the problem to be solved. The definition of the problem is the first and the most critical step of customer's SOP analysis. [17]

To develop huge number of software applications, every year these software industries are spending billion and trillions rupees. Further, it becomes very difficult to predict the success of software project because the scope of the project keeps changing depending upon the market; hence the resources have to be re-allocated leading to schedule slippage and cost overruns. Many software projects involve multiple entities such as companies, divisions, etc., that may have varying interests. There is often a feeling of disconnection between software developers and customer's SOP engineering team, each believing that the others are out of touch with reality resulting in misunderstanding and lack of communication that in turn leads to failures in software project.

The success rate of these software projects mainly depends on many factors. However, if we see current software market situation, statistically, 31% of projects are being cancelled before they ever are completed. 53% of projects cost twice as of their original estimates, overall, the success rate is less than 30% [2]. However, it has almost been a decade since the software industry has detonated. It has witnessed a remarkable growth and a tremendous escalation not only in the core activities but also in the IT enabled services. Despite the uninterrupted expansion, the software industry still has the highest number of project delays and failures. According to the Standish report [1], 44% of the software projects are challenged (late, over budget and/or with less than the required features and functions) and 24% have failed (cancelled prior to completion or delivered and never used). Thus, making a total of 68% (both challenged and failed) which is quite exponential. Boehm [2] found that 15-35% of all the software projects were cancelled outright while the remaining projects suffered either from schedule slippage, cost overruns or failure to meet the project goals. Why did the project fail? From symptoms to root cause -what are the major factors that cause software projects to fail? What are the key ingredients that can reduce project failure?

There are many factors and reasons due to which software projects fail like Lack of user involvement, Long or unrealistic time scale, Poor or No Customer's SOP, Inadequate Documentations, Scope Creep, No Change Control System, Poor testing, Lack of foresight in building efficiency markets, poor managerial decisions, Cost overrun, Lack of an experienced project manager, Lack of methodology in the process, inadequate well-defined Schedules etc. Among all these reasons if any of the reason occurs in project then it leads to failure of software project. These are the some higher level failure factors but there are many other hidden factors also responsible for the failure of software project. Hidden factors means they are part of Software Development Life Cycle project. Software Development Life Cycle (SDLC) is nothing but end to end work flow of software project life. SDLC has different phases like customer's SOP gathering, software designing, development, testing, maintenance etc. But we go through the functionality of these different phases of SDLC then these phases are also have many factors which leads to failure of software project. For example if we consider the customer's SOP gathering phase of SDLC, it has following number of factors like Failures of customer's SOP gathering process, Lack of Knowledge about the business context, Lack of Understanding of Business problems/opportunities, Missing of gaps to be bridged, Inadequate number of Resources, Inadequate Time etc. Along with these problems miscommunication between customer's Requirement engineering team and software development, testing team is one of the major factor for the project failure. Root cause of this miscommunication and knowledge gap between customer's SOP engineering team and other teams is the volatile nature of customer's SOP coming from end customers or clients. A customer's SOP in volatile nature is by default because we cannot skip the changing nature of Customer's demands. And here software developer and testing becomes slave to fulfill the volatile customer's SOP from the client. Miscommunication and knowledge gap etc factors are mainly responsible for the failure in customer's SOP gathering process and in turn lead to failure of software project. If we consider software design phase, designer may miss some important customer's SOP due to inadequate knowledge about the product and its impact to incomplete development of software project. Same case can happen in software development phase where developer can miss something important to implement due to knowledge gap and it lead to

incomplete or erroneous software product. Ultimately, all these miss impacts of software testing and maintenance phase and it leads to big failure in software project.

Considering such situation, there is a need to have error free software product so that rate of software project failure will reduce. For error free software product, software industry should mainly focus on failures present in each phase of SDLC process. Also in current era, recognition of such failures is manual process. There is need of an hour to automate such manual process so that software designer, developer and tester will have list of failure factors readily in hand. Moreover, based on these failure factors, each individual can take better action.

To give better solution for the above mentioned problems, in this research researcher mainly focused on the analysis of different factors that are responsible for the failure of software project. In addition, researcher has thrown light on the failure factor of each phase of SDLC process. Researcher has collected information from many different software companies to get better evidences for the analysis of such failure factors. Researcher also focused on failure of customer's SOP gathering process and its impact on software design, software development, software testing and maintenance phase. Out of the analysis of the different software failure factors, researcher has recommended better action to overcome in such situations. Researcher has strongly focused on the design of automatic model that help customer's SOP-engineering team to identify the problems in different software component design documents and will take actions to overcome such problems. This automatic model is not only helpful for customer's SOP engineering team but also helpful for software design, development, and testing team.

Impact analysis is a key aspect of responsible requirements management. It provides accurate understanding of the implications of a proposed change, which helps the team make informed business decisions about which proposals to approve. The analysis examines the proposed change to identify components that might have to be created, modified, or discarded and to estimate the effort associated with implementing the change. Skipping impact analysis doesn't change the size of the task. [28]

1.2. Research Problem

In today's IT world, more and more online Applications and tools are used, which help in carrying out various daily chores. If these applications and tools do not work according to specification then it would cause inconvenience to all users. Many failures are playing role as root cause behind the software not working properly. Failure of any software leads to minimization of productivity, revenue of the software industry. Software failure can occur in any stage of Software Development Life Cycle (SDLC). As we have seen, that SDLC contains many phases like Customer's SOP, Design, Development, Testing and Maintenance. These phases can have their own failures and success. If we consider Customer's SOP phase, customer's SOP collection is one of the major activity of customer's SOP phase and it is an early phase of the software development life cycle. In Customer's SOP gathering phase, customer's SOP are the foundation of the software development. They provide the basis for cost estimation and for planning project schedules as well as for designing and testing specifications. The success of software product, both functionally and financially is directly related to the quality of the customer's SOP. Although the initial sets of customer's SOP are well documented, but incorrect and incomplete customer's SOP will occur during the software development lifecycle. The reason behind the incomplete customer's SOP can be the constant changes (addition, deletion, modification) in customer's SOP gathering process during the development lifecycle. Such constant customer's SOP is known as Volatile customer's SOP. Customer's SOP in volatile nature are by default because we cannot skip the changing nature of Customer's demands. Moreover, here software developer and testing team becomes slave to fulfill the volatile customer's SOP from the client. It affects the cost, the schedule and the quality of final product.

In addition, the miscommunication between customers SOP engineer and customer, knowledge gap, inadequate knowledge about the software product and unavailability of supported infrastructure etc are others factors which impact on success rate of software project. The miscommunication between customer's SOP engineering team and other teams is very well known and common factor. It also affects the cost and schedule of the software project. This miscommunication happens from the customer's

SOP engineering team. But if we consider customer's SOP engineering is one of the very important phase of SDLC process and failure of this phase almost affects all the phases of SDLC process. Customer's SOP gathering process plays a driving role during the software product creation. In every software project development life cycle, customer's SOP gathering and analysis phase plays the most important role to precede further functionality of SDLC process. Stability of customer's SOP potentially makes an impact on the success of later phases in a software project, including the success of test cases. According to Brooks [12], the toughest part in building a software system is to decide precisely what customer's SOP to be developed. Furthermore, the poor customer's SOP gathering and analysis may affect negatively at a later stage [13, 14]. Moreover, predicting potential results of the later phases from early time of software development can obviously help the project team to better deal with the risks of project rescheduling and resulting in a low-quality product [15, 16].

As we have seen, failures of customer's SOP engineering affects the all the phases of SDLC process, software testing is one of well-known and impacted phase of SDLC due to failure of customer's SOP engineering phase. [4] In the development of large, software-intensive systems, the system's customer's SOP are seldom, if ever, concluded upon prior to commencing with systems development life cycle process. This research shows that, in order to manage development, testing and domain complexities, instances of customer's SOP engineering (RE) and systems testing (ST) processes tend to inter-weave. However, missing customer's SOP information can cause one to create (or recreate) the needed information during different software testing ST activities. While backtracking in the software development process is known to be costly, the costs associated with missing customer's SOP in the SDLC process have not been investigated empirically.

To overcome these problems there is need to understand the root cause of such problems and tricks to solve the problems. In this research, researcher mainly focused on understanding the root cause of software project failures by analysing different factors which may affect the success rate of software product. Also thrown light on how poor, incorrect and incomplete customer's SOP gathering process affects the other phases of

SDLC process. But while analysing these factors researcher came to know that this is manual process and to complete this work there is need to collect related data from experienced employees from different software companies. Also this process contains manual intervention of human and as we know the nature of humans, it may create so many mistakes and it strongly leads to failure of software project.

To overcome such manual intervention problem there is need to have generic, automatic model which will identify the gaps, problems in the customer customer's SOP and will provide the exact problem type so that software customer's SOP engineer can correct his/her mistake may be by involving customer or by himself/herself. Hence, researcher focused on analysis of impact of poor customer's SOP gathering process on SDLC process, which in turn affect the software testing process. In addition, researcher has provided design of the model, which will analyse the customer's SOP problems and will provide the exact issue type present in customer's SOP document just to make customer's SOP engineer to take correct action. The functionality of this model will be automatically and for that user or customer's SOP engineer just need to provide single customer's SOP document. To meet the customer customer's SOP specifications there is need to take error free customer's SOP from client. In the current state of software engineering, business analyst and designer use many customer's SOP gathering techniques but for error free customer's SOP, there is need to understand which customer's SOP gathering technique is correct and most suitable. Hence in this research, researcher has focused mainly on analysis of different customer's SOP gathering techniques and its impacts on software testing process. Along with different customer's SOP gathering techniques, researcher has also analysed different factors which may lead to failure of software customer's SOP gathering process. These factors have to consider taking correct actions by the customer's SOP engineer to make error free customer's SOP specifications and in turn to develop error free software product. Failure of customer's SOP gathering process always affects all the phases of SDLC process and hence in this research, researcher has thrown light on how poor, incorrect and incomplete customer's SOP gathering process affects the software testing process.

Karl E. Wiegers has mentioned following truths of software customer's SOP in his book "More About Software Customer's SOP: Thorny Issues and Practical Advice(Microsoft Press, 2006; ISBN 0-7356-2267-1)Chapter 2: Truths About Software Customer's SOP, which are considered as problem statement for the research topic , which could have impact on software testing.[23]

1. Truth #1: If you don't get the customer's SOP right, it doesn't matter how well you execute the rest of the project.
2. Truth #2: Customer's SOP development is a discovery and invention process, not just a collection process.
3. Truth #3: Change happens.
4. Truth #4: The interests of all the project stakeholders intersect in the customer's SOP process.
5. Truth #5: Customer involvement is the most critical contributor to software quality
6. Truth #6: The customer is not always right, but the customer always has a point.
7. Truth #7: The first question an analyst should ask about a proposed new customer's SOP is, "Is this customer's SOP in scope?"
8. Truth #8: Even the best customer's SOP document cannot— and should not—replace human dialogue
9. Truth #9: The customer's SOP might be vague, but the product will be specific.
10. Truth #10: You're never going to have perfect customer's SOP.

1.2.1 Focus of research and motivation

Literature shows that the collecting proper SOP from customer has impact on its business of software development process and the factors responsible for development process [5] [6]. Customer's SOP to SDLC transition is defined as the most severe among

different software information leaks between development phases [7]. Relative expense for fixing defects, introduced in customer's SOP, are three times higher if found during architecture phase and five to ten times higher if found during construction phase of development of the system [8]. But I could not find any empirical example of the impact of poor (missing and incomplete) customer's SOP on software testing process, in terms of effort, time etc. This motivates us to look into the impact of missing, incorrect and incomplete customer's SOP and customer's SOP' attributes' information during SDLC process and its impact on Software testing.

C. J. , Fuller, Tremblay, & Berndt found accurately capturing system customer's SOP is the major factor in the failure of 90% of large software projects," echoing earlier work by Lindquist who concluded "poor customer's SOP management can be attributed to 71 percent of software projects that fail.[25]

Without a clear understanding of the problem, the tasks of formulating and prioritizing solutions are likely to lead to wrong conclusions. Wrong conclusions result in unnecessary solutions to wrong or non-existent problems. The resulting software products are unmarketable and/or unusable.

Although there are some differences, the common element to any definition of customer's SOP analysis is the understanding of the user problem that the proposed software project aims to resolve. As an example, Peter Horan defines customer's SOP analysis in terms of problem solving, and indicates that problem solving may be viewed as a sequence of steps. A neat acronym, attributed to Bransford8, outlines five steps involved in solving a problem, the first two of which

- Identify the problem and define the problem - constitute customer's SOP analysis, as shown in the following steps, while the last three usually involve customer's SOP management [18]

1. Identifying the problem to be solved
2. Defining the problem
3. Exploring alternatives
4. Acting on. Selected solution methods
5. Learning about the outcome of the chosen method

Karl E. Wiegers defines customer's SOP as "a specification of what should be implemented.

[Customer's SOP] are descriptions of how the system should behave, or of a system property or attribute. They may be a constraint on the development process of the system.[19]

Ralph Young describes the goals of customer's SOP analysis as identifying incorrectly elicited assumptions, ensuring consistency, increasing compliance, reducing misunderstandings between organizations and individuals, improving the responsiveness of suppliers, improving the satisfaction of all customers, writing good customer's SOP, and emerging the real Customer's SOP. [21]

1.3 Basic terms used in Research Topic [Annexure-II]

Definitions

1) **Customer's SOP**

   The software customer's SOP are description of features and functionalities of the target system. Customer's SOP convey the expectations of users from the software product. The customer's SOP can be obvious or hidden, known or unknown, expected or unexpected from client's point of view.

   A customer's SOP is a capability to which a project outcome (product or service) should conform. The purpose of customer's SOP management is to organize customer's SOP documents and allow the users to view and/or edit the customer's SOP. Customer's SOP management begins with the analysis and elicitation of the objectives and constraints of the software system. [24]

   A software customer's SOP specification is a description of a software system to be developed. It lays out functional and non-functional customer's SOP.

   Software customer's SOP specification establishes the basis for an agreement between customers and contractors or suppliers (in market-driven projects, these roles may be played by the marketing and development divisions) on what the software product is to do as well as what it is not expected to do. Software customer's SOP specification permits a rigorous assessment of customer's SOP

before design can begin and reduces later redesign. It should also provide a realistic basis for estimating product costs, risks, and schedules. [26]

2) Customer's SOP Analysis

- Customer's SOP analysis in systems engineering and software engineering, encompasses those tasks that go into determining the customer's SOP or conditions to meet for a new or altered product or project, taking account of the possibly conflicting customer's SOP of the various stakeholders, analysing, documenting, validating and managing software or system customer's SOP[22]
- Customer's SOP analysis is important because "If you do not have the correct Customer's SOP, you cannot design or build the correct product, and consequently the product does not enable the users to do their work [20]

3) Customer's SOP Analyst Responsibilities

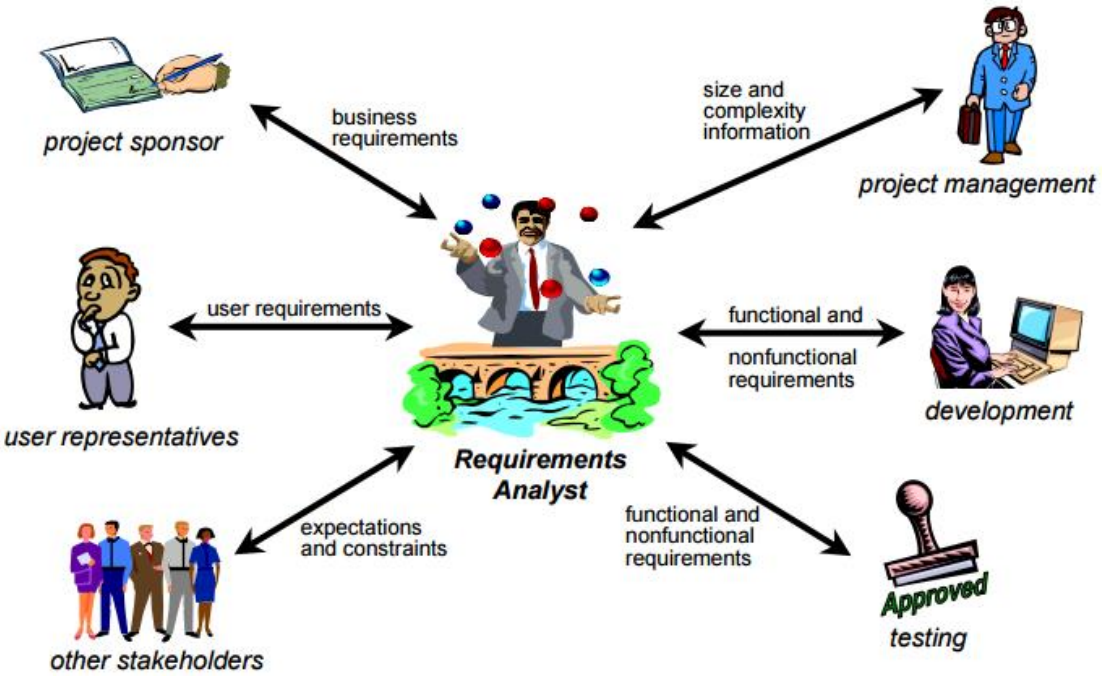


Fig. 1.2 Customer's SOP Analyst Role.

- Understand and **help the project manager** create the project's business case by making sure all the high level customer's SOP are listed in the project scope

- **Conduct a cost-benefit analysis** to justify the feasibility of the proposed solution. In this analysis, a comparison is made between the money that is being spent on the project and the benefits obtained from it. Obviously, it must be studied whether the project will be profitable before committing any time and resources for it.
- **Identify the involved stakeholders** by *either* getting the stakeholders list from the initial stakeholders *or* conducting a study to analyze who all will be involved/affected by the project. Sometimes, a combination of both these techniques may also be used.
- **Gather customer's SOP** from the key stakeholders by using customer's SOP elicitation techniques like brainstorming, customer's SOP workshops, focus groups and others.
- **Validate customer's SOP** by cross-referencing then with other stakeholders and try to get a buy in. It's important to achieve a consensus against the customer's SOP before actually start building the solution.
- **Analyze and interpret customer's SOP** for their viability against the business objective. The last thing any client will want is 'a fully functional product/service that is not able to solve the problem the project was built for'.
- **Recommend workarounds**, value additions and remove solution bottlenecks for the stakeholders. Since a Business Customer's SOP Analyst is having knowledge of both business and technology, he is able to propose solutions other might not think of.
- **Document customer's SOP** by creating use cases, functional and customer's SOP specifications documents. Also, Categorize customer's SOP as functional (contains the features required by the end-users), non-functional (customer's SOP for the performance and usability of the project) operational (operations that are carried out in the background) and technical and accordingly segregate them in different types of documents
- Not all customer's SOP may be important and feasible considering the scope and schedule of a project and thus these customer's SOP customer's SOP to be managed and prioritized by the Business Customer's SOP Analyst by working closely with the business owners
- **Prototype and model customer's SOP** – An important step towards letting the end users 'feel' what they might get at the end of the project completion. Also, prototyping aids in solution verification, error identification and getting an early feedback regarding the user interface of the project.

- **Gain customer's SOP sign-off** from the key stakeholders by making sure all of them are on the same level of understanding against the customer's SOP and then getting a written approval from them on the customer's SOP to be developed.
- Overall, **be the Customer's SOP Gatekeeper** and validate any new customer's SOP for their effect and impact on the existing customer's SOP set of the project.
- **Aid in development and testing** of the product – Business Customer's SOP Analysts are frequently seen doing the unit testing of the features of their projects and are also assisting the testing team in test case preparation
- **Be a part of the Change Control Board** – An optional responsibility where in an event of any change to the customer's SOP/s, the Business Customer's SOP Analysts is required to first assess the need of the change, deduce the impact of the change to the complete project, propose any workarounds if possible and then along with other board members (usually the Business representative, PM, technical lead) collectively decide whether to go ahead with the change or not.
- **Prepare end user documentation/manuals** – An optional task which the Business Customer's SOP Analyst may have to do in case a formal Technical Writer is not available
- **Give client presentations**, show and tell sessions, Overview sessions – Again, an optional task which customer's SOP to be covered by a Business Customer's SOP Analyst in case the Project Manager or Project Coordinator is not available

4) **Software-** Software means computer instructions or data. Anything that can be stored electronically is software. Software is a collection of instructions that enable the user to interact with a computer, its hardware, or perform tasks.

5) Software Testing [Annexure – II]

Software Testing is evaluation of the software against customer's SOP gathered from users and system specifications. Testing is conducted at the phase level in software development life cycle or at module level in program code. Software testing comprises of Validation and Verification.

Software testing describes the process of interacting with a piece of software with the aim of revealing errors. The actual type of software to be examined has a large influence on how this testing is performed.

Software Testing Basics

Software testing is performed as a means of validation and verification. Validation describes the process of determining whether the right software is built. For this, customer's SOP documents describe what the software is supposed to do. In contrast, verification describes the process of determining whether the software is built right, i.e., if it is correct with regard to its design. As software testing cannot be exhaustive in general it can never show the absence of errors. Therefore, the aim of testing is to detect as many errors in the software as possible.

Common terms such as error, fault, failure, bug or defect are sometimes inconsistently used. The available literature is full of different definitions of such terms.

A **failure** is defined as a deviation of the software from its expected delivery or service, while the cause of such a failure is a fault (Grindal and Lindström, 2002). The term error is used synonymously to fault.

A fault can have many different faces. For example, Goodenough and Gerhart (1975) distinguish between performance and logical errors, where the former is a problem of wrong timing and the latter is a problem of wrong functionality. Goodenough and Gerhard further suggest distinction between different kinds of logical errors, whether an implementation does not conform to a specification whether a specification does not correctly represent the design, whether a design does not fulfil the customer's SOP, and so on. A well-known, detailed taxonomy of faults is given in Beizer's classical book on software testing (Beizer, 1990). Each of these error types can manifest in different ways.

Some faults are more complex than others, and the effects of some faults might be more severe than the effects of other faults. Testing usually does not have to focus on one specific kind of fault, because the coupling effect (DeMillo et al., 1978) states that complex faults are linked to simpler faults. Consequently, it is sufficient to test for simple faults in order to detect complex faults.

1.4. Proposed Solution for Problem

1.4.1. Data collection and Analysis:

Thus in this research, Survey has been conducted where investigated at what extent poor customer's SOP impacts during the SDLC process and impact of that poor (missing/incorrect) customer's SOP on Software testing process. The main focus behind to conduct the survey is to collect real and actual information from the experienced people of different software companies. After collecting all information, researcher has done pilot study on collected data to check the reliability of the data. Using Cronbach's Alpha technique reliability of the data is 0.713. As result is 0.713 it proves that collected data is reliable.

In this survey, data has been collected based on gender, education, work experience in terms of total number of years, designation as a to collect personal front information. Research has been conducted on the employees who work in company residing in PMC and PCMC areas of Pune city. For these employees, researcher has targeted to collect General Background information of each employee. In term of general background, researcher has considered so many points like Designer and Tester of different software companies with respect to their Gender, Age, Education, Occupation and office location in pune city.

After general back ground information, researcher has focused on current customer's SOP gathering process that is getting executed in software industry. Under this section, researcher has collected information like from whom and how customer's SOP analyst or designer collects customer's SOP. And which kind of customer's SOP they are collecting in how much duration? Based on employees response researcher has analyzed the types of customer's SOP gathering ways and also analyzed the time duration required to gather customer's SOP from client or customer. Along with time duration and type of process gathering techniques, researcher also collected information related to the involvement of different people in customer's SOP gathering process and interaction with end user while doing customer's SOP gathering. In this section researcher mainly focused on different type of individuals from organization like designer, developer, tester, manager among them who actually frequently getting involved in the customer's SOP gathering process and gathering actual customer's SOP from client. After getting actual

responsible person for customer's SOP gathering, which kind of customer's SOP documents are useful for business analyst or designer or customer's SOP engineers like functional customer's SOP document, customer customer's SOP document, business customer's SOP document, component design specification etc.

This kind of documents always helps to all the teams of software industry. Functional Customer's SOP Document (FRD) is most important document as per most of respondents. As FRD is created for the functionality of proposed software and it is useful for Testing, development and designer team as well. Hence, most preference has been given to this document. FRD always contains the detail information about integrated functionality of proposed software and customer's demands or customer's SOP in the technical language.

Hence, FRD should always be in sync with updated customer's SOP and updated functionality of domain. Component design document (CDD) actually contains the information about technical, business functionality developed component of proposed software, and hence this document is mainly useful for development and testing team. Testing team uses CDD document just to understand business functionality in technical terms. Component Specification Document (CSD) is needed for designer and development team and hence employees from both these teams have given preference to this document. Test Case Document (TCD) is mainly important document for testing team and hence this document should be in sync with FRD that is the business functionality of proposed document. Customer Customer's SOP Document (CRD) is the base for all documents.

This document actually created by customer itself and accordingly business analyst and designer creates BRD, FRD, and other documents. CRD always contains customer's specific demands or customer's SOP and it is available in the customer's understanding language. BRD is created by considering CRD and it contains only business functionality of proposed software. It does not have technical functionality and hence this document is only helpful for business analyst and system architecture team.

There are different kinds of customer's SOP gathering techniques like Personally Meeting, Through Documents, and Online-Automated etc. Means using any of these three customer's SOP gathering techniques, business analyst and designer can collect

error free customer's SOP from client. Personally Meeting with client is the best customer's SOP gathering technique as it is nothing but face to face communication with client and using this technique business analyst can easily clarify all the doubts regarding customer's SOP specification.

By using "Through Document" technique, business analyst can get customer's SOP in written and hence in future there will not be misunderstanding between client and business analyst.

Online-Automated technique is useful when shared server is available for client as well as software product provider. Therefore, that in one place all customer's SOP can be stored and client as well business analyst can access it at any time.

For correct resource allocation and management, researcher also collected the information about the Business Analyst's time consumption on non-customer's SOP gathering activities. In many organizations, it has been seen that due to workload, business analyst need to work on non-customer's SOP gathering activities as well.

Hence, due to this extra workload, business analyst could not focus on his/her actual customer's SOP gathering activities, which in turn leads to incorrect, incomplete customer's SOP.

Following are the main activities that designer / customer's SOP engineer does as non-customer's SOP activity

- Writing Customer's SOP Documents
- Reviewing FRD/BRD
- Client Customer Interaction and Conducting Training for Testers and Developers etc.

After collecting customer's SOP gathering related information, researcher mainly focused on due to which reasons software project gets failed and hence information has been collected on this base line.

To analyze why software projects are getting failed, researcher has to consider following factors

- Lack of user involvement
- Long or unrealistic time scale
- Poor or No Customer's SOP

- Inadequate Documentations
- Scope Creep
- No Change Control System
- Poor testing
- Lack of foresight in building efficiency markets
- poor managerial decisions
- Cost overrun
- Lack of an experienced project manager
- Well-defined Schedules etc.

After analyzing different factors responsible, which lead to failure of software project, researcher has focused on erroneous or failed software project. Because non-qualitative software project can also lead to failure of software and this plays important role in the qualitative functionality.

To analyze the quality of product software testing plays major role in software industry. In addition, in Software industry, testing is a branch where verification of software's functionality is happening.

Once development team complete their development and submit product to testing team, then testing team start verification of functionality of software via test cases execution. However, most of time software has too much issues or errors because of many factors. Hence, it is quite important to understand which factors are responsible to make software erroneous.

In this research following list of factors are considered

- Logic Design
- Documentation
- Human
- Environment
- Data
- Interface
- Customer's SOP Errors etc.

These factors are mainly responsible to create errors in software project. Hence, in this research, researcher has recommended to overcome on these issues so that the rate of software project failure will be reduced.

As we know that Customer's SOP gathering is the process to collect demands, customer's SOP, or customer's SOP from client. After collecting such customer's SOP, business analyst customer's SOP to store and manage these collected customer's SOP. In software industry, many tools are available to collect record and manage customer's customer's SOP. For this research, following number of customer's SOP gathering tools have been considered and analyzed.

1. Visual Paradigms
2. Project Management Software
3. Microsoft-Package
4. Data Dictionary
5. Use Cases and User Stories
6. ReqHarbor
7. MindTool
8. IBM Rational Doors

To major the effect of customer's SOP gathering process's failure, researcher has also consider the involvement of different people in the customer's SOP gathering process. Software tester is the main focus in the analysis of people involvement in customer's SOP gathering process. As Customer's SOP gathering covers many pros and cons like if customer's SOP is error free then there is lots of possibility of success of software project but if anything miss by business analyst then it leads to error in customer's SOP. Only person might be miss few important points or alternation while collecting customer's SOP.

There is big possibility of lack knowledge about existing system or product and that lead to incorrect or incomplete customer's SOP collection from client. Hence, instead of only one person like business analyst is not enough for correct customer's SOP collection.

There is need to involve many people like Senior management team, senior architecture team, testers, developers, clients, end users and subscribers in the discussion of customer's SOP gathering meeting or session. Because if any person misses any

important point or not from the customer's SOP gathering discussion then another might catch that point and likewise correct and complete customer's SOP can get collected from client. Also due to involvement of senior management, project management, testers and developers all the aspects of system (to be developed) are getting covered and considered. After studying and analyzing different factors customer's SOP gathering process, researcher has focused on different factors of software testing process. Because the main aim of this research is to analyze the impact of incorrect/incomplete customer's SOP gathering process on software testing process.

Software Testing is a process used to help identify the correctness, completeness and quality of developed computer software. Testing is a process of executing a program with the intent of finding an error. [8] Testing is a process rather than a single activity. This process starts from test planning then designing test cases, preparing for execution and evaluating status till the test closure.[9]

There are two types of software testing process has been considered i.e. manual and automated testing. In manual testing, tester customer's SOP to create test case, test data and manually execute test cases with dummy data on particular software component but in case of automated testing, test cases and dummy data has been created by testing tool itself. From the analysis of these two types of testing researcher has recommended automated testing is most useful testing in many software companies and other software tester should use the automated testing to increase the productivity of software project.

Along with different types of software testing types, researcher has also considered whether tester are using software testing tool or not. Many software testers responded to use testing tools to make software testing process easy.

How testers are utilizing their complete daytime for test cases execution. As we know that automated testing is quite easier, faster and increase the productivity of testing team, test cases execution per day by using automated testing also increases the count of testing.

But using manual testing as it is manual process and requires lots of human intervention, hence, test cases execution using manual testing giving quite low count.

Hence in this research survey has been carried out mainly on the test cases execution per day using automated testing mode. Total number of test case execution per day also plays important role in maintaining quality if software. As per 1 resource 4 test cases are

executed per day which is quite low number of agreed view of employee. But maximum around 360 employees are agreeing that up to 8 test cases can be executed per day. And yes it is correct count because 8 test cases executed per day it is standard count of software testing. So as per survey it is recommended that execution of 8 test cases per day using automated testing mode is best practice for software testing process. But in case of testing scenarios are easy or there is urgency from client then in that case tester can test up to 16 or 20 test cases in one day.

Testing team uses different documents to verify the product functionality and hence in this research, researcher has analyzed the significance of different documents like

- Customer Customer's SOP Document (CRD),
- Business Customer's SOP Document (BRD),
- Functional Customer's SOP Document (FRD),
- Component Specification Document (CSD),
- Component Design Document (CDD),
- Test Case Document (TCD) etc.

To analyze the impact of poor customer's SOP gathering and analysis process there is need to know how productivity of software testing team can be increased and what are the hurdles that can be minimizes the success rate of software testing phase of SDLC process.

Hence in this research, researcher has considered cost factor which is most important factor for the failure or success of software project.

Cost is being calculated based on the following list of factors i.e. Resources, software, Hardware, Network, Infrastructure (electricity, rent etc.).

From the analysis researcher has concluded that Hardware, resources are two important factors that software testing customer's SOP to focus because these two factors affects success rate of software project.

Finally yet importantly, to complete the main objective of this research, researcher has focused on different factors of customer's SOP gathering process that are actually affecting the software testing process. As we know that Poor customer's SOP gathering is nothing but issues present in the collected customer's SOP from client or customer.

A poor customer's SOP is nothing but erroneous customer's SOP. As we saw in the section of failure of software process, poor customer's SOP always affect complete

SDLC cycle and hence in software testing point of view, there is need to understand what kind of work actually getting affected mainly in software testing team.

Software testing team mainly deal with following list of tasks:

- Addition of test case
- Deletion of test case
- Modification of test case
- Re-execution of test case
- Verification of newly added functionality due to customer's SOP change
- Test results creation for newly added customer's SOP etc.

If there is change in customer's SOP or customer's SOP is incorrect then it might affect to the complete list of above tasks. Test case creation, review and update in test cases is basic activity of software testing team.

While writing test cases, test team follows Functional Customer's SOP Document (FRD) and as we know that FRD is nothing but one of the customer's SOP document created by customer's SOP team. But if FRD is incorrect or incomplete then test cases created by testing team can also be incorrect because incorrect FRD obviously lead to incorrect or incomplete test cases. Incorrect or incomplete customer's SOP means poor customer's SOP and if customer's SOP is poor then definitely it affects on the work of software testing team.

Along with different factors of customer's SOP gathering process, which actually leads to failure in software testing process, researcher also studied and analyzed different overheads of software testing process due to poor customer's SOP gathering process. Due to issues present in customer's SOP, many overheads can occur in software testing process.

For this research, researcher has considered following list of overheads:

- GAP in Testing,
- Increase in System Failures,
- System Testing Delay,
- Inaccurate Testing Estimation,
- Test Team Credibility,
- Delay Benefit Realization.

An overhead is the extra burden on testing team and it always decreases the productivity of testing team. Above listed overheads are mostly found as factors which affects software testing productivity. Hence, it is quite important to understand which overhead is faced by testing team most. After analyzing different overheads researcher has recommended that Gap in Testing is the major overhead that software testing team is facing due incorrect customer's SOP gathering process.

Software testing process also gets impacted due to some common issues present in customer's SOP gathering process. Software is developed according to Clients Customer's SOP. Here some customer's SOP issues are discussed with software developer and tester, which may affect software-testing process.

Design or customer's SOP issues can occur in any phase of SDLC process. The possibility of rework due to customer's SOP issues or defect can be minimum if these issues are getting solved in customer's SOP or development phase itself but if customer's SOP issues comes in testing phase then it affects testing, customer's SOP and development phase as well.

In this research, following parameters has been considered as software testing issues like

- Absence and Incompleteness,
- Incorrectness, Ambiguity and Vagueness,
- Volatility, Traceability etc.

After analyzing these issues, researcher has recommended that 'Absence and Incompleteness' is the major factor which leads to failure of software testing process.

1.4.2. Model Design for Customer's SOP Management to Reduce Software Failures

Considering the present state of impact of poor customer's SOP gathering process on software testing, a model to reduce software failure in testing phase need to design through the present research work. This new model is designed named as Customer's SOP Management to Reduce Software Failures (RMRSF). The main functionality of RMRSF model is to provide better software testing actions for corresponding poor customer's SOP. Model has 3 phases like input, processing and output [9, 10]. RMRSF model integrates the functionality of different modules like Input module, processing module and output module

1.4.2.1.Input Phase

Input phase will contain input module which deals with input data collection from the end user and this input data will be in the form of functional customer's SOP document (FRD) or Customer Customer's SOP Document (CRD). In this phase, Customer's SOP document can get from your local computer drive. Main functionality of this phase is to get exact type and customer's SOP document and call the processing module for further analysis.

1.4.2.2.Processing Phase

Processing phase deals with integrated functionality of Reading Customer's SOP document, analyzing customer's SOP document by call Customer's SOP Management Engine (RA Engine) and Execution of Output module to generate list of Issues, Impacts and Actions for particular type of customer's SOP issue. Processing phase contains processing module which will deal with integrated functionality of Reading Customer's SOP document, analyzing customer's SOP document by call Customer's SOP Management Engine (RA Engine) and Execution of Output module to generate list of Issues, Impacts and Actions for particular type of customer's SOP issue.

In processing phase, RM Engine gets call to analyze exact requirment issue type. Customer's SOP Analysis Engine (RA Engine) will be the major part of processing module and will be developed based on the customer's SOP issues responded by respondent from different software companies. For RM Engine, following customer's SOP issues will be considered [11].

1. Incomplete/Absent
2. Incorrect
3. Ambiguity & Vagueness
4. Volatility
5. Traceability

RAEngine is heart of RMRSF module. Without RAEngine, RMRSF can not do anything. RAEngine is basically works on if else ladder concept. It first checks what is exact requirment issue present in provided requirment document and based on that decide type of customer's SOP issue. For deciding appropriate customer's SOP issue, RAEngine

analyze requirment document by compairing it with software system architechure document and tries to provide exact customer's SOP issue. RAEngine takes Requirment document as an input and generates customer's SOP issue type by considering many isssues present in provided customer's SOP document.

As metioned above, RAEngine maninly focuses on five type of requirment issues like incomplete,incorrect, Ambuiguity,vaguess, volatility and traceability etc. [4]. Based on different conditions like if track table is missing or improper change control process found then its mark customer's SOP issue as traceability issue. If functional or non-functional customer's SOP are missing then it marks requirment issue type as incorrect. If there is change between old requirment document and new customer's SOP document then it marks requriement issue as volatility. If customer's SOP followed poor customer's SOP definitoin then it marks customer's SOP issue as Ambiguity and Vagueness. Like wise it checks for Incomplete/Absence customer's SOP issue. Here using if syntax RAEngine verifies many conditions to decides appropriate requirment issue.

Once requirment issue is identified by RAEngine, it returns that customer's SOP issue back to processing module and then processing module works on further analysis.

1.4.2.3. Output generation phase

Output phase deals with generation of output based on input argument as a customer's SOP issue type provided by processing phase. Output phase has module named as Output Module and this module is basically gets executed by Processing module. Main functionality of output module is to get customer's SOP type issue as an input and based on this input query to database to fetch corresponding list of Issues, Impacts and Action points. This module is displaying list of Issues, Impacts and Actions based on corresponding customer's SOP issue type.

Output module deals with database to fetch records from three different tables named as Issue, Impact and Action. The perquisite of this model is these tables should get created with data in database.

1.5. Thesis Organization

Chapter 2 discusses the importance, scope, objectives and hypothesis of the study. It also describes the research methodology and research design with primary and secondary data collection. Along with Data collection, it also explains different data collection methods. It also contains the limitation of the study and chapter schemes.

Chapter 3 discusses the relevant background literature and research gap.

Chapter 4 presents the data analysis, results and interpretations.

Chapter 5 Gives Conclusion and Suggestions which provides model with detail design Customer's SOP Management to Reduce Software Failures Model (NMRSF)

This chapter deals with introduction about the research. It explains the research problems in details. Also provide the proposed solution for the research problem. It also explains the motivation behind this research. It also listed the list of chapters this thesis will have.

Reference:

1. The Standish Group Report (CHAOS). (2003). Retrieved November 2013, from :http://www.projectsmart.co.uk/docs/chaos-report.pdf.
2. Boehm, B., & Bose, P. (1994). A collaborative spiral software process model based on Theory W. Third International Conference on the Software Process, pp. (59-68).
3. Cao, L., & Ramesh, B. (2008). Agile Customer's SOP Engineering Practices: An Empirical Study. IEEE Software, 25 (1), pp. (60-67).
4. 2013, Md Rounok Salehin "Missing Customer's SOP Information and its Impact on Software Architectures:A Case Study" The School of Graduate and Postdoctoral Studies The University of Western Ontario,London, Ontario, Canada
5. Gross, A., Doerr, J. (2012). What do software architects expect from customer's SOP specifications? results of initial explorative studies. IEEE First International Workshop on Twin Peaks of Customer's SOP and Architecture, IEEE Software, pp. (41-45).

6. Lee, S., & Rine, D. (2004). Missing Customer's SOP and Relationship Discovery through Proxy Viewpoints Model. 19th annual ACM Symposium on Applied Computing, pp. (1513-1518). Nicosia, Cyprus.
7. George, B., Bohner, S. A., & Prieto-Diaz, R. (2004). Software information leaks: A complexity perspective. Ninth IEEE International Conference on Engineering Complex Computer Systems (ICECCS'04), pp. (239-248). Florence, Italy: IEEE Computer Society.
8. Gumuskaya, H. (2005). Core Issues Affecting Software Architecture in Enterprise Projects. Proceedings of World Academy of Science, Engineering And Technology, volume 9, pp. (35-41)
9. Thomas Kiihne , "What is Model?" Darmstadt University of Technology, Darmstadt, Germany.
10. http://www.la1.psu.edu/cas/jgastil/pdfs/conceptualdefinitiondeliberation.pdf
11. Kirsten Kiefer, "The Impact of Customer's SOP issues on testing", Software Education associates Ltd
12. Brooks, F. 1987. No Silver Bullet: Essence and Accidents of Software Engineering. IEEE Computer, Vol. 20, No. 4, April 1987, 10-19.
13. Jayaswal, B. K., Patton, P. C. 2006. Design for Trustworthy Software: Tools, Techniques, and Methodology of Developing Robust Software, 1st Edition. (September 2006), Prentice Hall edition.
14. Zowghi, D. 2002. A Study on the Impact of Customer's SOP Volatility on Software Project Performance. Proceedings of Ninth Asia-Pacific SE Conference (APSEC‟ 2002), IEEE Computer Science.
15. Taghi, M., Khoshgoftaar, N., Sundaresh, N. 2006. An empirical study of predicting software faults with case-based reasoning. (June 2006), Software Quality Control.
16. Jiang, Y., Cukic, B., Ma, Y. 2008. Techniques for evaluating fault prediction models. (October 2008), Empirical Software Engineering.
17. Vera Berenbaum The Effect of software customer's SOP analysis on project success and product quality in Rochester Institute of Technology RIT Scholar Works
18. Bransford, J. D. and Stein, B. S., The IDEAL Problem Solver, Freeman, 1984.

19. Wiegers, Karl E., "When Telepathy Won't Do: Customer's SOP Engineering Key Practices", Process Impact, www.processimpact.com/articles/telepathv.html [Wiegers attributes the definition to Ian Sommerville and Pete Sawery, Customer's SOP Engineering: A Good Practice Guide. Wiley, 1997.]
20. Suzanne and James Robertson, Mastering the Customer's SOP Process. ACM Press, 1999
21. Ralph R. Young, Effective Customer's SOP Practices. Addison-Wesley, 2001, page 108.
22. Kotonya, G. and Sommerville, I. 1998. Customer's SOP Engineering: Processes and Techniques Chichester, UK: John Wiley and Sons.
23. Karl E. Wiegers More About Software Customer's SOP: Thorny Issues and Practical Advice(Microsoft Press, 2006; ISBN 0-7356-2267-1)Chapter 2: Truths About Software Customer's SOP
24. Henry Johnson, An approach to software project management through customer's SOP engineering, At Texas Tech University, Henry Johnson, December 2010
25. Davis , C.J, Fuller, R.M. Tremblay, M.C. & Berndt, D.J. (2006). Communication Challenges in customer's SOP elicitation and the use of the repertory grid technique. Journal of computer information Systems, 78
26. Bourque, P.; Fairley, R.E. (2014). "Guide to the Software Engineering Body of Knowledge (SWEBOK)". IEEE Computer Society. Retrieved 17 July 2014
27. http://technosoftwares.com/software-development-life-cycle
28. http://www.jamasoftware.com/blog/change-impact-analysis-2

Chapter 2

Research Design and Methodology

2.1 Introduction

In chapter 1 introduction of title is explained and frequently used basic terminology has been explained. Chapter 1 gives information about research problem, proposed solution of research problem, Suggested Model for problem.

The study is related to the Collecting needs from customer which is termed as SOP for software product development. Survey based research methodology has been used to carry out this research. Data collection for this research is done using purposive and convenience sampling methods. This is out of necessity because it was almost impossible to obtain a perfect random sample. PMC and PCMC study area has been considered in the scope of this research. Software companies reside in Hinjewadi, Kharadi, Viman Nagar, EION IT Park, SB Road area visited to collect sample data. It was not possible to identify the perfect sample population, due to the unavailability of complete and reliable data about. The method of selection of the sample is described in this chapter and after that the nature of primary data and secondary data is explained.

2.2 Statement of the Problem

A good set of needs of SOP are the base for any software development process. Collecting needs from customer is playing main role to estimate cost and schedule as well as developing design and testing specifications. [1, 2] Hence quality of needs playing

main role in the success of any software project. Even though Customer's SOP needs are freezed in initial phase of software project but it may got change throughout the software development lifecycle. Change in need means it can be addition, deletion or modification. Such kind of change in need during SDLC (Software Development Life Cycle) [9] always impacts the cost, schedule and quality of software product [2]. The reason to fail any software product is mainly depends upon the quality of collected needs. Hence, a good set of customer's needs mentioned in SOP are needed for any software project, to be successful. But if customer's needs are not specified clearly, correctly against what the system should do, then many projects will fail in this case. In fact, many systems have just been given a deadline for delivery, a budget to spend, and a vague notion of what it should do.

2.3 Importance of the Study

To make life easy there is need of automation and automation is possible only due to computerization of most of electronic appliances. Computerization is nothing but integration of hardware and software. Software plays most important role in the automation in most of electronic appliances. Hence, demand for qualitative softwares increasing highly to make electronic appliances. However, some pitfalls may lead to create failures in softwares. Collecting needs from customer process is one of the major factor to create failures in software development. Collecting needs from customer process is the base for software development lifecycle (SDLC) process. To develop any software there is need to have correct and complete customer needs then only quality software product will produce. Quality of Software or Software Quality is a term deals with

verification of developed software and developed software should meet the customer satisfaction [7]. Most of developers from MNCs make their primary goal to produce qualitative system that meets the needs of the user. Hence, to develop quality software, developers should focus on correct and complete needs from customers. Along with correct and complete needs, verification and validation of software product should be mandatory. Validation and verification of software product functionality is being done by testing team. Tester from Software Testing should also concentrate on qualitative customer needs. If customer needs are incorrect, incomplete then it is definitely going to impact of Software development process and business also.

Following are the few characteristics of needs related points which may responsible for the failure of software product.

1. Absent and incomplete requirement
2. Incorrect Requirements
3. Ambiguity and vagueness
4. Volatility
5. Traceability

If customer's need is incomplete or important part is absent in requirement document then it might lead to failure in all the phases of SDLC. Incomplete customer's need always lead to incomplete functionality and delay in project delivery to client. The reason behind incomplete customer's need is delay from customer or business analyst may have lack of product knowledge. Hence as per most of respondent Incomplete or absent customer's need is the root cause of software failure. Hence, researcher of this

research recommends that customer's need should be complete or should not be missing any important part.

Volatility is nothing but changes in customer's need. Volatility mainly happens due to changes in customer's demands or lack of product knowledge of requirement analyst. Customer's need plays base role for all phases like development, testing etc. Because development team works on the base of requirement document but if, requirement document frequently is changed then development team need to rework according to changes in requirement document. Changes in customer's need also affect the testing phase as well. If customer's need gets changed then testing team also need to retest all the test cases and need to verify updated functionality. Most of time testing team needs to add test cases as well for updated customer's need. Hence, updated customer's need always lead to rework for development and testing team. In turn, volatility leads to increase the workload for development and testing team, and it leads to decrease productivity of both teams. If productivity decreases then it increases failure in software productivity.

Incorrectness is also important factor in customer's need issues. Incorrectness generally happens due to inadequate knowledge of business analyst. Business analyst does not understand complete functionality of software product and hence they cannot map customer's demands and software product functionality. Most of time, Incorrectness occurs due to incomplete of customer's need or missing or absent of important part in requirement document. Incorrect customer's need always lead to development and verification of incorrect functionality and in turn it affects to productivity of development and testing team.

Ambiguity and vagueness issue occur due to lack of product knowledge. Most of business analyst having less experience and hence they don't aware about the complete functionality of software product. Hence, they can not map customers' demands with functionality of their software product. Due to lack of product knowledge, customer's need becomes vague and ambiguice. But if customer's need becomes vague and ambiguice then development and testing team needs to consume more time on their work. It requires more time for customer's need understanding and implementation. Testing team also need to consume more time on customer's need understanding and test cases creation.

If customer's need is volatile then there is need to keep track of each and every change for the betterment of SDLC process. So that testing and development team can verify component functionality as per latest changes made in requirement document. To keep such changes there is need to have common place so that requirement, development and testing team can have easy access of this place. In software engineering such place is know as traceability. Traceability always gets updated by requirement team if requirement gets change. But if any change is getting miss in the traceability sheet then it might lead to incompleteness of customer's need and as we saw above, incompleteness of customer's need leads to failure of software product.

2.4 Scope of the Study

The study is related to the collecting needs from customer for software development process and its impact on business of IT companies. Pune city has been considered for this

research work. As this research mainly focused on collecting needs from customer for software development process and its impact on business of IT companies.

The scope of this research is software companies resides in

1. PMC area
2. PCMC area

Pune is the second largest city in Maharashtra and well known for educational facilities, research institutes and software industry. Due to the good educational facilities, Pune is called as "**The Oxford of the East**" and hence students from all over the world are getting attracted towards pune city. Due to big software industry, pune is transforming into vibrant modern city with bubbling activities in the IT and Hi-Tech sectors. Thousands of software companies can be found in pune city. And as there are software development industries, SDLC process surely gets followed by all software companies.

During the course of the present study the researcher has focused on collecting needs from customer for software development process and its impact on business of IT companies.. Also it is focused on provision of model which will help to reduce the failures of software product due to customer's needs collecting process. This model has been designed by considering parameters like cost, time etc. The researcher has also done analysis of current scenarios of software development process and tools used in software industries.

The geographical location of Pune city and software companies present in PMC and PCMC are indicated by the map 2.1, 2.2 and 2.3 as follows.

Map 2.1 Map of Pune city

Map 2.2 Map of the software companies present in PMC area

Map 2.3 Map of the software companies present in PCMC area

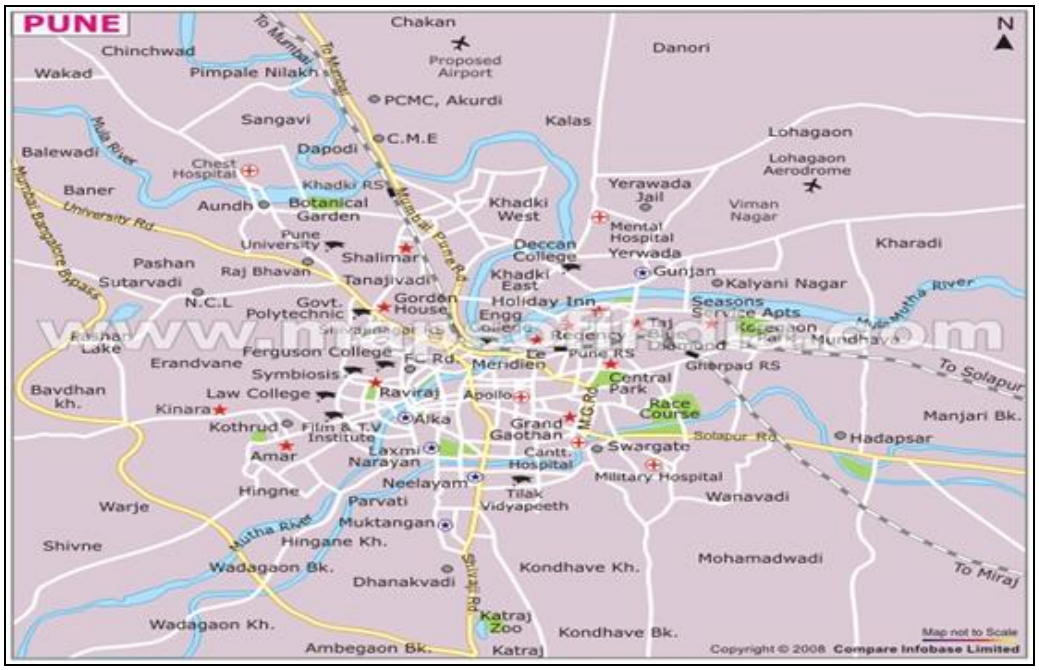


Figure 2.1.Source: http://www.mapsofindia.com/maps/maharashtra/pune.htm (23/7/2008) **[3]**

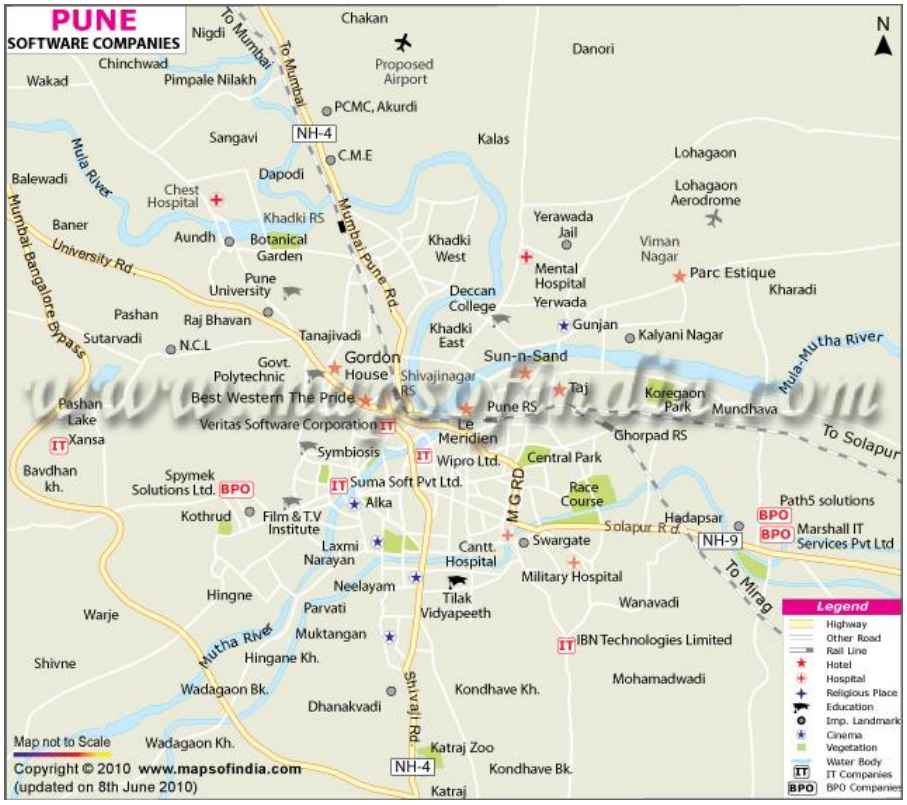


Figure 2.2 : http://www.mapsofindia.com/pune/software-company-pune.html[4]

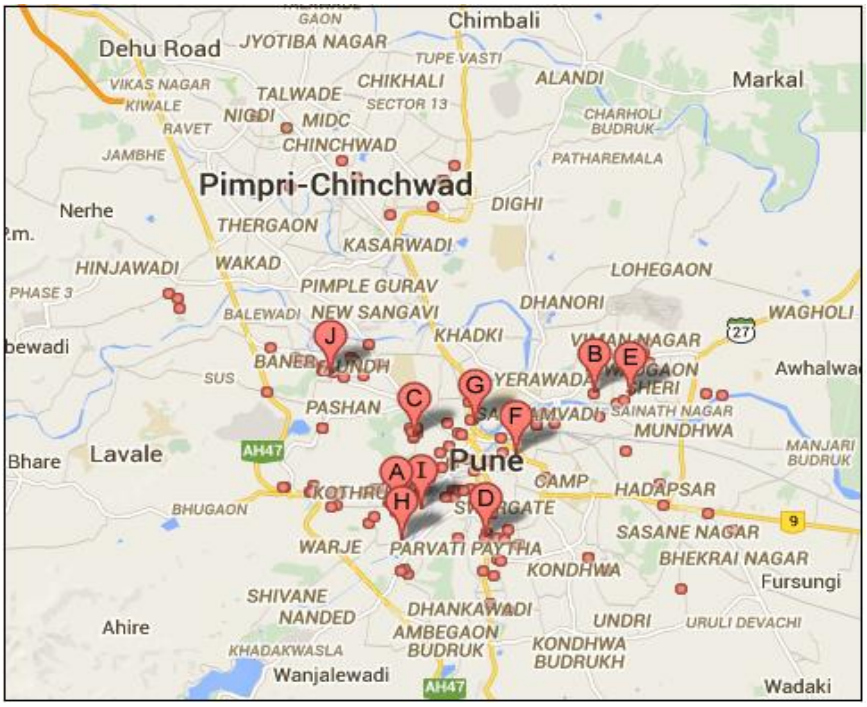


Figure 2.3 :Source : https://maps.google.co.in/maps?hl=en-IN&gbv=2&ie=UTF-8&fb=1&gl=in&q=software+companies+in+pune&hq=software+companies&hnear=0x3bc2bf2e67461101:0x828d43bf9d9ee343,Pune,+Maharashtra&ei=av_nVKH6O86IuwSvrIL4Bw&ved=0CB4QtQM&output=classic&dg=brw[5]

2.5 Objectives of the study

To study collecting needs process in software development and its impact on business of software development. With this main objective, the other objectives are as follows:

1. To study various task undertaken for software development process in IT Companies.
2. To Study the various tools and techniques used in collecting initial needs for the software product development.

3. To identify various factors responsible for the software development.
4. To study impact of Collecting needs from customer on business of IT companies.
5. To draw conclusion and suggestions.

2.6 Hypothesis of the Study

In consistent with the objectives, the researcher formed following hypotheses:

Hypothesis 1: *There are hurdles in collecting customer needs in software development.*

Hypothesis 2*: IT Industry follows standard practices to use licensed or well-known tools to collect initial needs from customer in software development.*

Hypothesis 3*: If collected needs are not freezed, then it has impact on business.*

2.7. Research Methodology

The researcher has used survey based research methodology to carry out this research. The study is related to verify the impact of poor requirement analysis on software testing. The researcher has considered the Pune and PCMC area for the study. This study is primarily focused on awareness of various tools used during development of software and problems face by testers in software companies in Pune and PCMC.s that is why primary data was collected from employees of Software Company in Pune. Researcher has used interview and questionnaire technique for data collection. Researcher has collected data from software companies from Hinjewadi, Magarpatta (Hadapsar) ,Shivaji Nagar and Kharadi.

Type of Industry	**Total Companies**	**5 % Sample of Companies**
Software Companies	424	21

Table No.2.1 Software Companies

Sr. No.	**Company Name**	**No. of Employees**
1	Accenture	7
2	Amdocs	24
3	Atos	46
4	Davachi	7
5	BMC	6
6	Capegemini	53
7	Citi Bank	6
8	Congnizant	9
9	Hummingbird	9
10	Calsoft	16
11	Neptune Inf Tech	2
12	IBM	4
13	KPIT Cummins	38
14	Patani	7

15	Persistent	9
16	Principal Optima	4
17	CLSA	7
18	Sigma Soft	51
19	Symphony	51
20	Tech Mahindra	34
21	Wipro	10

Table No. 2.2 Number of Employees Company wise

By applying purposive sampling, Total 21 companies have been identified for study which has more than 250 cr. Turnover. [93-98]

By applying Quota sampling, Researcher has divided respondents in 3 categories Business Analyst, Designer, Testers.

This research study is related to study the collecting needs from customer for software development process. It utilizes both primary and secondary data. The secondary data utilizes already available information both published as well as unpublished. For primary data however such a facility is not available and it has to be collected by using the survey method. The scope of research is limited; the survey is undertaken by obtaining a purposive and quota sample. The description of the research methodology required for the process of obtaining a sample as well as the nature and size of sample should be adequately explained. Purposive and convenience sampling techniques involves the selection of respondents based on the important characteristics under study such as where

they work, position in organization, specific knowledge related to the research problem etc.

To study various factors affected if client SOP (Statement of Purpose) keep on changing during software development Process, researcher has collected data from 10 Companies. From each company 5 clients data has been gathered for measuring impact if client is asking for changes in SOP frequently during software development process.

Sr.No	**Company Name**
1	KPIT Cummins
2	SAP
3	TechHighway
4	Harmony
5	Intelizign
6	L & T Infotech
7	ATOS
8	Zensar
9	CLSA
10	Davachi

Table No. 2.3 List of Companies selected for analyzing SOP

2.7.1 Primary data

Primary data are obtained through a survey. Such data is first hand and original in nature. Several methods are used for collecting primary data like telephone survey/e-mail survey, mail questionnaire, personal observation and interviews. Particularly in survey, the important ones are – observation, interview, questionnaire, schedules, e-mail survey, telephone survey etc. Each method has its advantages and disadvantages. The primary data collected by the researcher is explained in the following manner:-

2.7.1.1 Selection of the city

For the present research work Purposive sampling method has used to select the Pune city as Universe of the study. Pune city is also known as “The Oxford of the East" and a center of IT activity. In this research, Pune city is defined as a scope for the study of impact poor requirement gathering on software testing and designing of model to reduce software product failures.

The type of research is **Exploratory Design** in which the **Survey Method** used for data collection; the focus is given to the aspects of tester’s problem in software companies.

2.7.1.2 Universe of the Study

For the present study Software companies located at Pune and PCMC area has been treated as a universe of study by using purposive sampling method.

2.7.1.3 Unit of the Study

21 Software Companies considered a sample for conducting review to understand the basic problems and technical problems while testing the software.

2.7.1.4 Sampling procedure:

The study units (Hinjewadi , Hadapsar, Kharadi, ShivajiNagar) approximately covers 400 testers.

2.7.1.4 Sample size and Groups:

As per Krejcie and Morgan's law(1970) if population is in between 75,000 and upto 10,00,000 then 384 sample size should considered so here in research researcher has considered it as 400.

In this research, 400 samples are collected from Business Analyst, Designer and Tester from different software companies they reside under PMC and PCMC area. They are shown in Table No.2.3

Sr. No.	Constituents	Number of Sample points in the sample from employees of software companies
1	Employees	400

Table No. 2.4: Selection of Sample

2.7.1.5 Parameters of Development:

- **Data collection :**

Primary Data Collection: Primary data for various samples has been collected in the following ways:-

a. Information has been collected from Business Analyst, Designer, and testers through the Questionnaire and interview schedule.

b. Data also has been collected through personal field visits to companies and focus group discussions with end users, who are using software for various uses.

Secondary Data Collection: The secondary data will be collected from books, journal articles and websites, newspapers and conferences souvenir.

- **Data Analysis**

The collected data has been analyzed by quantitative and qualitative ways. The SPSS (Statistical Package for Social Science Research) is used for quantitative analysis.

2.7.2. Secondary Data

The Secondary data is used to study the awareness and usage of SDLC models and impact of software requirement gathering process on software testing with the help of earlier research studies made by others. It is also used to find out the merits and demerits and limitations of different SDLC models and awareness of collecting needs from customer's process with the help of available data. It is helpful to study the objectives and hypotheses framed for the present study.

The secondary data is collected from reputed journals and magazines, newspapers, articles, internet websites and archives. For collecting this data the researcher has visited various libraries. A few of these libraries are Jaykar Library (SPPU University), Yashada, British Library, Indira College of Science Library and Indsearch Library.

2.8 Statistical tools used for this research

For the current research, primary data has collected by visiting software companies. As explained above secondary data has been collected from existing journals, books etc. For this research, primary and secondary data has been analyzed with the help

of software like Microsoft Excel, SPSS (Statistical Package for Social Science) with version 19.0. Statistical techniques like Percentage, Average, Cross tabulation, the techniques of hypotheses testing etc. are also used. Charts and graphs are also prepared and used to support the analysis of the data wherever necessary.

2.9 Time Budgeting

Duration from 2012 to 2014 is considered for a study collecting needs from customer's for software development process and its impact on business

2.10 Limitation of the Study

The scope of research is limited to the data published and sources available from different software companies located in PMC and PCMC area of Pune City. Technical details of the system do not covered in the scope of this research. Most of part of this research is carried out based on secondary data. While selecting secondary data for model designing to reduce the software product failures, the data found as scattered in different software companies. Total number of population of properties and population in respect of each software company can not be exactly ascertained. However, an attempt to locate the maximum number of employees from different software companies itself. The study pertains to only PMC and PCMC area of Pune City.

2.11 Chapter Scheme

The chapter scheme for this thesis is as follows:-

Chapter No.	Name of Chapter
1	Introduction
2	Research Design and Methodology

3	Review of Literature
4	Data Analysis and Interpretation
5	Conclusions and Suggestions
	Appendices
	Bibliography

The first chapter is the introduction where the researcher has given a brief background about the study.

The second chapter, Research Design and Methodology, has discussed the importance, scope, objectives and hypothesis of the study. It also describes the research methodology and research design.

The third chapter deals with the Review of Literature. It describes the review of the existing available literature on the collecting needs process, awareness of different SDLC models, and software testing process. It gives an insight into the history of the software requirement engineering process, software testing process, SDLC model and impact of collecting needs process on software development business.

The fourth chapter presents the analysis of the data for Software testers, developers and Designers. This chapter deals with the testing of hypothesis.

The fifth chapter provides detail design of model for the betterment of software development process including enhancement of collecting needs process and summarizes observations, conclusions, findings and suggestions of the present study.

References have been given at the end of each chapter themselves and a selected bibliography is given at the end.

This chapter discusses the importance, scope, objectives and hypothesis of the study. It also describes the research methodology and research design with primary and secondary data collection. It has explained different data collection methods. It also contains the limitation of the study and chapter schemes.

Reference:

[1] M.P.Singh, Rajnish Vyas, "Requirements Volatility in Software Development Process" International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-2, Issue-4, September 2012

[2] Zowghi, N. Nurmuliani, ―A study of the Impact of requirements volatility on Software Project Performance, Proceedings of the Ninth Asia-Pacific Software Engineering Conference , APSEC 2002, Gold Cost, Queensland, Australia,04-06 Dec 2002, pp:3-11.

[3] http://www.mapsofindia.com/maps/maharashtra/pune.htm (23/7/2008)

[4] http://www.mapsofindia.com/pune/software-company-pune.html

[5] https://maps.google.co.in/maps?hl=en-IN&gbv=2&ie=UTF-8&fb=1&gl=in&q=software+companies+in+pune&hq=software+companies&hnear=0x3bc2bf2e67461101:0x828d43bf9d9ee343,Pune,+Maharashtra&ei=av_nVKH

6O86IuwSvrIL4Bw&ved=0CB4QtQM&output=classic&dg=brw

[6] Sr. S. P. Gupta, "Statistical Methods", Sultanchand & Sons Publication, New Delhi.

[7] Don Gotterbarn, "Reducing Software Failures: Addressing the Ethical Risks of the Software Development Lifecycle" Australian Journal of Information Systems.

[8] Research Methodology Methods and Techniques By C R Kothari and Gaurav Garg. Pg 52-109

[9] http://technosoftwares.com/software-development-life-cycle

Chapter 3

Review of Literature

3.1. Introduction

In second chapter research methodology has been explained and data collection and sampling method is also explained in detail.

In this chapter an extensive literature review has been done on the concepts and theories related to the implication of software testing and requirement engineering. A review of research papers and articles has been undertaken to take note of and acknowledge work that has been don e in this field. The researcher has collected secondary data from reputed journals and magazines, newspapers, articles, internet websites and archives. The researcher has visited libraries in and around Pune City, to collect secondary data. The researcher has identified research papers published in renowned journals and conference proceedings along with articles published in newspapers on various topics such as implementation of impact of poor requirement gathering process on SDLC phases and in turn on software testing process etc. The review of available literature on each topic is taken into account in this chapter.

The researcher has done a literature review on each and every criteria of software Testing and impact of requirement engineering. These criteria focus mainly on various aspects of Software requirement gathering, design, development, testing and software maintenance like-

- Definition of Software
- Software Engineering Process
- Software Development Life Cycle
- Software Development Life Cycle Models
- Software Requirement
- Requirement Engineering
- Software Testing
- The Testing Spectrum

- Type of Software Testing
- Impact of Poor requirement gathering process on Software Testing Process

3.2. Definition of Software

1. **Muhammad Naeem Ahmed Khan** et.al., have published research paper on "**Review of Requirements Management Issues in Software Development**" [1]

Muhammad Naeem Ahmed Khan and their friends have defined software as it is more than just a program code. A program is an executable code, which serves some computational purpose. Software is considered to be collection of executable programming code, associated libraries and documentations. Software, when made for a specific requirement is called software product.

3.3. Software Engineering Process

Along with definition of software, **Muhammad Naeem Ahmed Khan** has also defined the process of software engineering. In his research article he said that software requirement engineering is all about developing products, using well-defined, scientific principles and methods [1].

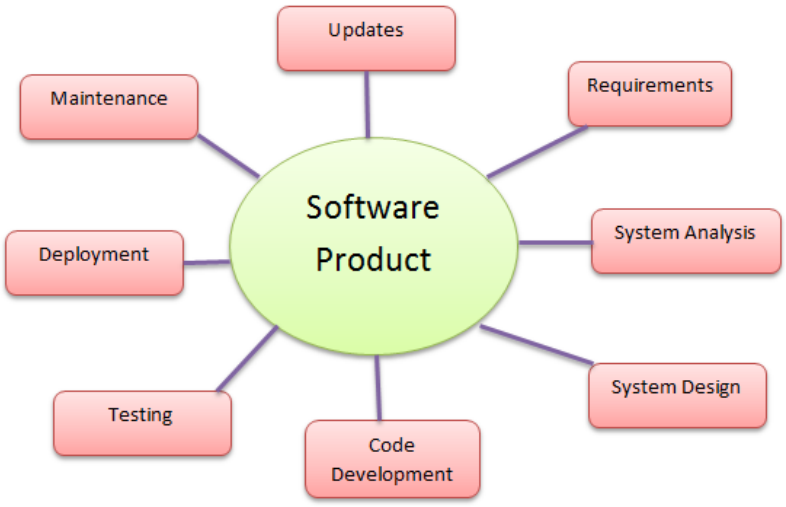


Figure 3.1: Software Engineering Process

Software engineering is an engineering branch associated with development of software product using well-defined scientific principles, methods and procedures. Software engineering is a branch of software development management.

2. **Hawkey** has published the second chapter on **Software Requirement Engineering** in **SEBackground**[2].

As per Hawkey, software engineering process is the model chosen for managing the creation of software from initial customer inception to the release of the finished product. The chosen process usually involves techniques such as[2].

- Analysis,
- Design,
- Coding,
- Testing and
- Maintenance

3.4. Software Development Life Cycle

3. Bender RPT has published his article on **"Systems Development Lifecycle: Objectives and Requirements. 2003".** [3]

Bender RPT glossary described in his chapter that Software Development Life Cycle (SDLC) is a process of building or maintaining software systems. Software development life cycle is the most important element in software development. It depicts the necessary phases in software development.

Software Development Life Cycle includes various phases from preliminary development analysis to post-development software testing and evaluation. It also consists of the models and methodologies that development teams use to develop the software systems, which the methodologies form the framework for planning and controlling the entire development process.

3.4.1. Phases of Software Development Life Cycle (SDLC)

4. Vanshika Rastogi has published her article on "**Software Development Life Cycle Models- Comparison, Consequences.2015**" [4]

Vanshika Rastogi has defined the Software Development Life Cycle (SDLC) as it is a framework that is used to understand and develop information systems and software successfully. It is a process used by almost all developers and software development

companies as the standard in the software process development. SDLC has many models and each model has its own strengths, weaknesses, advantages and disadvantages [4]. Software Development Life Cycle (SDLC) is a process used by software industry to design, develop and test high quality softwares. The SDLC aims to produce high quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates.

Vanshika Rastogi also mentioned the following six phases are present in every Software development life cycle model [4]:

1. Requirement gathering and analysis
2. Design
3. Implementation or coding
4. Testing
5. Deployment
6. Maintenance

1) Requirement gathering and analysis: Business requirements are gathered in this phase. This phase is the main focus of the project managers and stake holders. Meetings with managers, stake holders and users are held in order to determine the requirements like; who is going to use the system? How will they use the system? What data should be input into the system? What data should be output by the system? These are general questions that get answered during a requirements gathering phase. After requirement gathering these requirements are analyzed for their validity and the possibility of incorporating the requirements in the system to be development is also studied. Finally, a Requirement Specification document is created which serves the purpose of guideline for the next phase of the model.

2) Design: In this phase the system and software design is prepared from the requirement specifications which were studied in the first phase. System Design helps in specifying hardware and system requirements and also helps in defining overall system architecture. The system design specifications serve as input for the next phase of the model.

3) Implementation / Coding: On receiving system design documents, the work is divided in modules/units and actual coding is started. Since, in this phase the code is produced so it is the main focus for the developer. This is the longest phase of the software development life cycle.

4) Testing: After the code is developed it is tested against the requirements to make sure that the product is actually solving the needs addressed and gathered during the requirements phase. During this phase unit testing, integration testing, system testing, acceptance testing are done.

5) Deployment: After successful testing the product is delivered / deployed to the customer for their use.

6) Maintenance: Once when the customers starts using the developed system then the actual problems comes up and needs to be solved from time to time. This process where the care is taken for the developed product is known as maintenance.

3.5. Software Development Life Cycle Models

5. Yogi Berra has published his article on **"Software Development Life Cycle (SDLC)"** [5]

Yogi Berra defines about (software/system) life cycle model is a description of the sequence of activities carried out in an SE project, and the relative order of these activities.

There are various software development approaches defined and designed which are used/employed during development process of software, these approaches are also referred as "Software Development Process Models" (e.g. Waterfall model, incremental model, V-model, iterative model, etc.). Each process model follows a particular life cycle in order to ensure success in process of software development.

6. Ms. Shikha maheshwari and Prof.Dinesh Ch. Jain 2012 have published article on **"A Comparative Analysis of Different types of Models in Software Development Life Cycle" [6]**

Ms. Shikha maheshwari and Prof.Dinesh Ch. Jain have descried different Software life cycle models phases of the software life cycle and the order in which those phases are executed. Each phase produces deliverables required by the next phase in the life cycle. Requirements are translated into design. Code is produced according to the design which is called development phase. After coding and development the testing verifies the deliverable of the implementation phase against requirements.

Ms. Shikha maheshwari and Prof.Dinesh Ch. Jain are also provided information about various software development life cycle models definition and desgin which are followed during software development process. These models are also referred as "Software Development Process Models". Each process model follows a Series of steps unique to its type, in order to ensure success in process of software development.

Following are the most important and popular SDLC models followed in the industry:

1. Waterfall Model
2. Iterative Model
3. Spiral Model
4. V-Shape model
5. Big Bang Model

3.5.1. The Waterfall Model

7. **Dr. Winston Royce** mentioned in his article **"Managing the development of larger systems"[7]**

In 1970 Royce introduce the waterfall model. It is the classic life cycle model. It is widely known, understood and used. In some respect, waterfall is the"commonsense" approach. Waterfall model is the simplest model of software development paradigm. It says the all the phases of SDLC will function one after another in linear manner. That is, when the first phase is finished then only the second phase will start and so on.

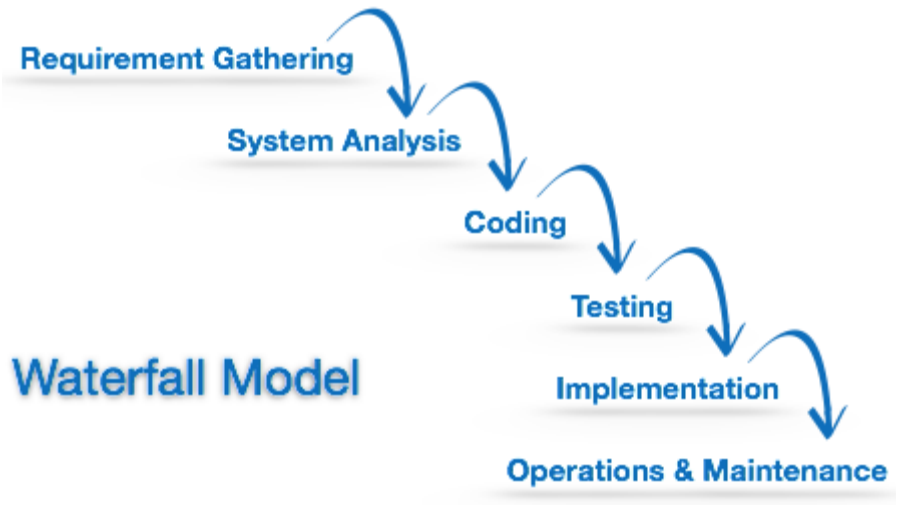


Figure 3.2: Waterfall Model

This model assumes that everything is carried out and taken place perfectly as planned in the previous stage and there is no need to think about the past issues that may arise in the next phase. This model does not work smoothly if there are some issues left at the previous step. The sequential nature of model does not allow us go back and undo or redo our actions.

This model is best suited when developers already have designed and developed similar software in the past and is aware of all its domains.

The sequential phases in Waterfall model are:

- **Requirement Gathering and analysis:** All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification doc.
- **System Design:** The requirement specifications from first phase are studied in this phase and system design is prepared. System Design helps in specifying hardware and system requirements and also helps in defining overall system architecture.
- **Implementation:** With inputs from system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality which is referred to as Unit Testing.

- **Integration and Testing:** All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.

- **Deployment of system:** Once the functional and non functional testing is done, the product is deployed in the customer environment or released into the market.

- **Maintenance:** There are some issues which come up in the client environment. To fix those issues patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment

3.5.2. Iterative Model

8. PK.Ragunath, S.Velmourougan, P. Davachelvan, S.Kayalvizhi, R.Ravimohan have written article on **"Evolving A New Model (SDLC Model-2010) For Software Development Life Cycle (SDLC)"** [8]

PK.Ragunath, S.Velmourougan, P. Davachelvan, ,S.Kayalvizhi, R.Ravimohan have describer iterative model as this model leads the software development process in iterations. It projects the process of development in cyclic manner repeating every step after every cycle of SDLC process [8].

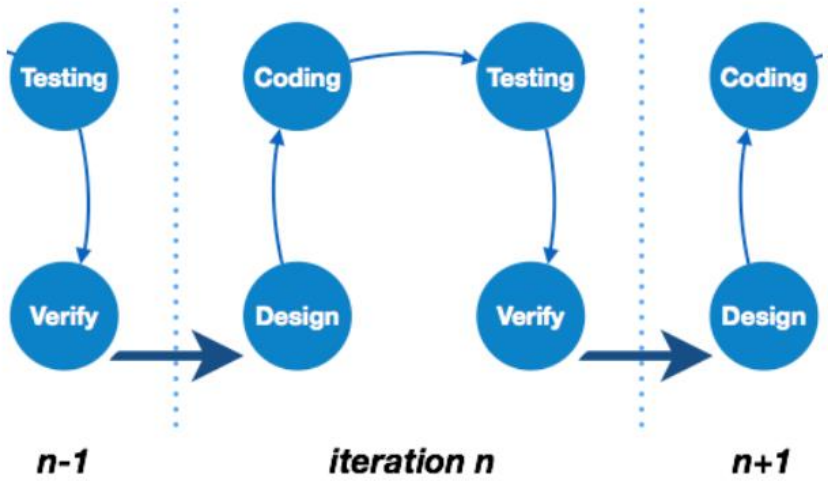


Figure 3.3: Iterative Model

The software is first developed on very small scale and all the steps are followed which are taken into consideration. Then, on every next iteration, more features and modules are designed, coded, tested and added to the software. Every cycle produces software, which is complete in itself and has more features and capabilities than that of the previous one.

After each iteration, the management team can do work on risk management and prepare for the next iteration. Because a cycle includes small portion of whole software process, it is easier to manage the development process but it consumes more resources.

Iterative process starts with a simple implementation of a subset of the software requirements and iteratively enhances the evolving versions until the full system is implemented. At each iteration, design modifications are made and new functional capabilities are added. The basic idea behind this method is to develop a system through repeated cycles (iterative) and in smaller portions at a time (incremental).

Iterative and Incremental development is a combination of both iterative design or iterative method and incremental build model for development. "During software development, more than one iteration of the software development cycle may be in progress at the same time." and "This process may be described as an "evolutionary acquisition" or "incremental build" approach."

In incremental model the whole requirement is divided into various builds. During each iteration, the development module goes through the requirements, design, implementation and testing phases. Each subsequent release of the module adds function to the previous release. The process continues till the complete system is ready as per the requirement.

The key to successful use of an iterative software development lifecycle is rigorous validation of requirements, and verification & testing of each version of the software against those requirements within each cycle of the model. As the software evolves through successive cycles, tests have to be repeated and extended to verify each version of the software.

3.5.3. Spiral Model

9. Seema, Sona Malhotra 2012 have published article on **Analysis and tabular comparison of popular SDLC models [9].**

Seema and Sona Malhotra have defined Spiral model as it is a combination of both, iterative model and one of the SDLC model. It can be seen as if you choose one SDLC model and combine it with cyclic process (iterative model).

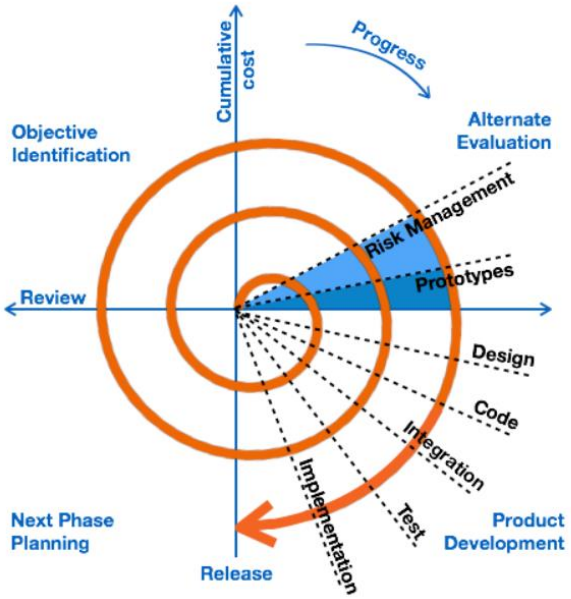


Figure 3.4: Spiral Model

This model considers risk, which often goes un-noticed by most other models. The model starts with determining objectives and constraints of the software at the start of one iteration. Next phase is of prototyping the software. This includes risk analysis. Then one standard SDLC model is used to build the software. In the fourth phase of the plan of next iteration is prepared.

The spiral model has four phases. A software project repeatedly passes through these phases in iterations called Spirals.

- **Identification:** This phase starts with gathering the business requirements in the baseline spiral. In the subsequent spirals as the product matures, identification of system requirements, subsystem requirements and unit requirements are all done in this phase.

This also includes understanding the system requirements by continuous communication between the customer and the system analyst. At the end of the spiral the product is deployed in the identified market.

- **Design:** Design phase starts with the conceptual design in the baseline spiral and involves architectural design, logical design of modules, physical product design and final design in the subsequent spirals.

- **Construct or Build:** Construct phase refers to production of the actual software product at every spiral. In the baseline spiral when the product is just thought of and the design is being developed a POC (Proof of Concept) is developed in this phase to get customer feedback.

  Then in the subsequent spirals with higher clarity on requirements and design details a working model of the software called build is produced with a version number. These builds are sent to customer for feedback.

- **Evaluation and Risk Analysis:** Risk Analysis includes identifying, estimating, and monitoring technical feasibility and management risks, such as schedule slippage and cost overrun. After testing the build, at the end of first iteration, the customer evaluates the software and provides feedback.

Based on the customer evaluation, software development process enters into the next iteration and subsequently follows the linear approach to implement the feedback suggested by the customer. The process of iterations along the spiral continues throughout the life of the software

3.5.4. V – Model

10. Sonali Mathur and Shaily Malik (2012) have published article on "Advancements in the V-Model" [10]

Sonali and Shaily mentioned in their article about the major drawback of waterfall model is we move to the next stage only when the previous one is finished and there was no chance to go back if something is found wrong in later stages. They also mentioned that V-Model provides means of testing of software at each stage in reverse manner **[10]**.

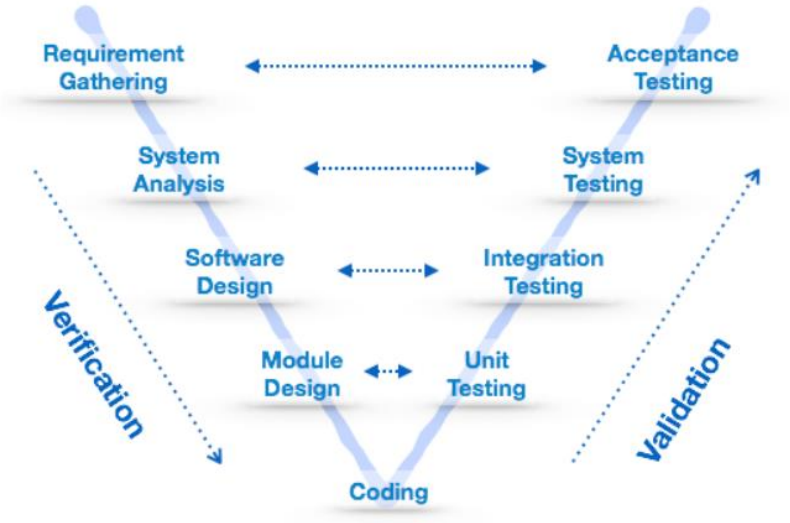


Figure 3.5: V- Model

V-Model provides means of testing of software at each stage in reverse manner.

At every stage, test plans and test cases are created to verify and validate the product according to the requirement of that stage. For example, in requirement gathering stage the test team prepares all the test cases in correspondence to the requirements. Later, when the product is developed and is ready for testing, test cases of this stage verify the software against its validity towards requirements at this stage.

Under V-Model, the corresponding testing phase of the development phase is planned in parallel. So there are Verification phases on one side of the .V. and Validation phases on the other side. Coding phase joins the two sides of the V-Model.

This makes both verification and validation go in parallel. This model is also known as verification and validation model. Hence in this research, V-model has been selected, studied and used for further research study.

3.5.4.1. Verification Phases

Following are the Verification phases in V-Model:

- **Business Requirement Analysis:** This is the first phase in the development cycle where the product requirements are understood from the customer perspective. This phase involves detailed communication with the customer to understand his expectations and exact requirement. This is a very important

activity and need to be managed well, as most of the customers are not sure about what exactly they need. The acceptance test design planning is done at this stage as business requirements can be used as an input for acceptance testing.

- **System Design:** Once you have the clear and detailed product requirements, it's time to design the complete system. System design would comprise of understanding and detailing the complete hardware and communication setup for the product under development. System test plan is developed based on the system design. Doing this at an earlier stage leaves more time for actual test execution later.

- **Architectural Design:** Architectural specifications are understood and designed in this phase. Usually more than one technical approach is proposed and based on the technical and financial feasibility the final decision is taken. System design is broken down further into modules taking up different functionality. This is also referred to as High Level Design (HLD).

  The data transfer and communication between the internal modules and with the outside world (other systems) is clearly understood and defined in this stage. With this information, integration tests can be designed and documented during this stage.

- **Module Design:** In this phase the detailed internal design for all the system modules is specified, referred to as Low Level Design (LLD). It is important that the design is compatible with the other modules in the system architecture and the other external systems. Unit tests are an essential part of any development process and helps eliminate the maximum faults and errors at a very early stage. Unit tests can be designed at this stage based on the internal module designs.

3.5.4.2. Coding Phase

The actual coding of the system modules designed in the design phase is taken up in the Coding phase. The best suitable programming language is decided based on the system and architectural requirements. The coding is performed based on the coding guidelines and standards. The code goes through numerous code reviews and is optimized for best performance before the final build is checked into the repository.

3.5.4.3. Validation Phases

Following are the Validation phases in V-Model:

- **Unit Testing:** Unit tests designed in the module design phase are executed on the code during this validation phase. Unit testing is the testing at code level and helps eliminate bugs at an early stage, though all defects cannot be uncovered by unit testing.

- **Integration Testing:** Integration testing is associated with the architectural design phase. Integration tests are performed to test the coexistence and communication of the internal modules within the system.

- **System Testing:** System testing is directly associated with the System design phase. System tests check the entire system functionality and the communication of the system under development with external systems. Most of the software and hardware compatibility issues can be uncovered during system test execution.

- **Acceptance Testing:** Acceptance testing is associated with the business requirement analysis phase and involves testing the product in user environment. Acceptance tests uncover the compatibility issues with the other systems available in the user environment. It also discovers the nonfunctional issues such as load and performance defects in the actual user environment.

3.5.5. Big Bang Model

11. Naresh Kumar, A. S. Zadgaonkar, Abhinav Shukla have published research article on "**Evolving a New Software Development Life Cycle Model SDLC-2013 with Client Satisfaction**" **[11]**.

Naresh Kumar, A. S. Zadgaonkar, Abhinav Shukla defined that Big Bang model is the simplest model in its form. It requires little planning, lots of programming and lots of funds. This model is conceptualized around the big bang of universe. As scientists say that after big bang lots of galaxies, planets and stars evolved just as an event. Likewise, if we put together lots of programming and funds, you may achieve the best software product **[11]**.

The Big Bang model is SDLC model where we do not follow any specific process. The development just starts with the required money and efforts as the input, and the output is the software developed which may or may not be as per customer requirement.

Big Bang Model is SDLC model where there is no formal development followed and very little planning is required. Even the customer is not sure about what exactly he wants and the requirements are implemented on the fly without much analysis.

Usually this model is followed for small projects where the development teams are very small.

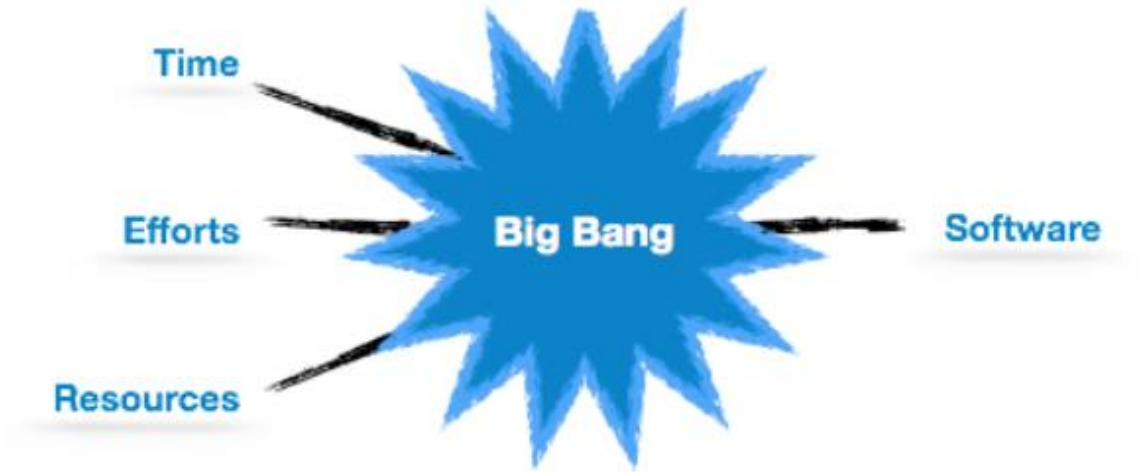


Figure 3.6: Big Bang Model

For this model, very small amount of planning is required. It does not follow any process, or at times the customer is not sure about the requirements and future needs. So the input requirements are arbitrary. This model is not suitable for large software projects but good one for learning and experimenting.

3.6. Software Requirement

12. G. Kotonya and I. Sommerville, (1998), have published article on "**Requirements Engineering: Processes and Techniques**", in the book published by Chichester, UK: John Wiley & Sons[12]..

G. **Kotonya and I. Sommerville** have given details information about how software requirements play a vital role in software development. They have defined a requirement is a statement of what a system is required to do and the constraints under which it is required to operate role that is baseline for every software project and having capability to which a system confirm to. Actually of the scope of customer work is defined by requirement only [1]. A requirement deals with objects or entities, the states they can be in, and the functions that are performed to change states or object characteristics [7].

3.7. Requirement Engineering

As we have seen earlier, **Muhammad Naeem Ahmed Khan** and et.all explained the definition of requirement engineering in his research paper. As per his paper, Requirement engineering is nothing but group of activities to elicit, analyze, specify, verify, validate and manage requirements **[1]**. Requirements engineering plays an important role in the software development projects.

13. Ramos Rowel and Kurts Alfeche (1998) have written chapter on "**Requirements Engineering A good practice guide**" John Wiley and Sons, 1998**[13]**

Ramos Rowel and Kurts Alfeche stated in the above mentioned book that requirement gathering is the practice of collecting the requirements of a system from users, customers and other stakeholders **[13]**.Requirements gathering practices include interviews, questionnaires, user observation, workshops, brainstorming, use cases, role-playing and prototyping.

3.7.1. Requirements Elicitation:

G. Kotonya and I. Sommerville have explained in Engineering processes and Techniques that requirement elicitation is first phase of requirements engineering process and its purpose is to discover requirements for the system being developed. Requirements

are elicited from customers, end-users and other stakeholders such as system developers [12].

3.7.2. Requirements Analysis and Specification:

14. **I. Sommerville and P. Sawyer (1997)**, Requirements Engineering: A Good Practice Guide, New York: John Wiley & Sons,.

I. Sommerville and P. Sawyer say in their book that requirements analysis is one of the first phases in requirements engineering and its purpose is to analyze the elicited requirements. Once the requirements have been gathered, then the conflicts, overlaps, omissions and inconsistencies need to be analyzed [14].

3.7.3. Requirements Specification

In this process, the requirements (both functional and non-functional) are documented. On the basis of the accumulated requirements, SRS document is created to which both the parties should agree upon.

3.7.4. Requirements Validation

This phase relates to the process of examining the requirements document to ensure that it pertains to the intended development of the right system (i.e., the system that the users expect) [12]. In this sense, validation processes purely commensurate to the functional requirements (FR).

3.7.5. Requirements Verification:

As per IEEE glossary[15], requirement engineering is the process by using any software company meets the customer's requirements, or needs and expectations.

16. K. E. Wiegers (2003) has published in the booked named as "**Software Requirements**", 2nd ed., Redmond, W A: Microsoft Press, 2003[16].

K. E. Wiegers says that Requirements engineering is the degree to which a system, component or process meets specified requirements (i.e., FR) as well as the

customer/user needs or expectations (i.e., NFR). It is the process of ensuring that the requirements' statements are accurate and complete as well as demonstrate the desired quality characteristics [16]. Hence, it pertains to non-functional requirements (NFR).

3.7.6. Requirements Management:

G. **Kotonya and I. Sommerville** have also explained about the Requirements management process and as per them it is the process of administers changes to the agreed requirements, relationships between requirements and dependences between the requirements document as well as other documents produced during the entire system and software engineering process [12].

3.8. Software Testing

Muhammad Naeem Ahmed Khan has defined the Software testing as it is a most often used technique for verifying and validating the quality of software [1].

17. Author has published detailed review of literature review in "**Literature Survey"** chapter.[17]

As per this chapter, Software testing is the procedure of executing a program or system with the intent of finding faults [17].

18. Donald Firesmith, has published article on "**Common Requirements Problems, Their Negative Consequences, and the Industry Best Practices to Help Solve Them**".[18]

As per **Donald Firesmith,** Software testing is the process of exercising or evaluating a system or system components by manual or automated means. It is used to validate specified requirements or to identify differences between expected and actual results. Testing is the measurement of software quality. The difficulty in software testing stems from the complexity of software field: A process and program cannot completely test with moderate complexity. The purpose of testing can be quality assurance, verification and validation, or reliability estimation. The complete testing is infeasible or impossible. As described earlier, there are two major areas of testing, i.e., correctness

testing and reliability testing. Correctness is the minimum requirement of software, the necessary purpose of testing. Correctness testing will require some type of database query, to tell the true behavior from the false one. As described in [18], Software reliability refers to the probability that software system will operate without failure and is related to main aspects of software including the testing process.

The main objective of software testing is to affirm the quality of software system by systematically testing the software in carefully controlled circumstances, another objective is to identify the completeness and correctness of the software, and finally it uncovers undiscovered errors. Software testing is measured to be labour intensive and expensive, which accounts for > 50 % of the total cost of software development [18]. Hi defined Software testing as it is a significant activity of the software development life cycle (SDLC). It helps in developing the confidence of a developer that a program does what it is intended to do so. In other words, we can say it's a process of executing a program with intends to find errors.

19. Olga Liskin , et al. 2012, have published article on "**Supporting Acceptance Testing in Distributed Software Projects with Integrated Feedback Systems: Experiences and Requirements**" [19]

Olga Liskin, et al have defined that in the language of Verification and Validation (V&V), black box testing is often used for validation (i.e. are we building the right software?) and white box testing is often used for verification (i.e. are we building the software right?) [19]]. In his research, his study emphasizes the need to investigate various testing techniques in software testing field; we have conducted a literature review to obtain the reviews from state-of-art.

3.9. The Testing Spectrum

20. **Vishawjyoti, Sachin Sharma**, have published research article on "Study and Analysis of automation testing techniques".

Vishawjyoti, Sachin Sharma says in his paper that software testing is involved in each stage of software life cycle, but the way of testing conducted at each stage of software development is different in nature and it has different objectives.

- **Unit testing** is a code based testing which is performed by developers, this testing is mainly done to test each and individual units separately. This unit testing can be done for small units of code or generally no larger than a class.
- **Integration testing** validates that two or more units or other integrations work together properly, and inclines to focus on the interfaces specified in low-level design.
- **System testing** reveals that the system works end-to-end in a production-like location to provide the business functions specified in the high-level design.
- **Acceptance testing** is conducted by business owners, the purpose of acceptance testing is to test whether the system does in fact, meet their business requirements.
- **Regression Testing** is the testing of software after changes has been made; this testing is done to make sure that the reliability of each software release, testing after changes has been made to ensure that changes did not introduce any new errors into the system.
- **Alpha Testing** Usually in the existence of the developer at the developer's site will be done.
- **Beta Testing** Done at the customer's site with no developer in site.
- **Functional Testing** is done for a finished application; this testing is to verify that it provides all of the behaviors required of it.

3.10. Type of Software Testing

21. **Antonia Bertolino** has published his research article on "**Software testing research and practice**"

Antonia Bertolino has defined Software as testing is classified based on each stage of software life cycle. This sections give details about different types software testing that are getting executed through out SDLC process. In this paper, Antonia Bertolino also mentioned the following types of software testing process.

- **Manual Testing**

22. Vivek Kumar (2012) has published his article on "**Comparison of Manual and automation testing**"

A test team is spending most of its time running test cases but is executing the tests slowly. It takes as much as a day just to test one new feature of a system, and often the tests fail due to system time-outs. Executing full regression tests has been so expensive that the team avoids doing so whenever possible. Needless to say, the test execution is manual.

- **Automated Testing**

23. R. M. Sharma (2014), has published his article on "**Quantitative Analysis of Automation and Manual Testing**" [23]

R. M. Sharma has defined Automation testing as, it is also known as Test Automation, is when the tester writes scripts and uses another software to test the product. This process involves automation of a manual process. Automation Testing is used to re-run the test scenarios that were performed manually, quickly, and repeatedly. Automated software testing is the best way to increase the effectiveness, efficiency and coverage of software testing.

- **Black box testing**

24. Harsh Bhasin, at.el. (2014), have published research article on "**Black Box Testing based on Requirement Analysis and Design Specifications**" [24]

Harsh Bhasin and his friends have given details of Black Box Testing which is also called functional testing. According to the strategy it does not need any knowledge of internal design or code structure etc. The author said that this strategy is totally based/focused on the testing for requirements and functionality of the work product/software application. Black Box testing strategy is totally based on external structure of the code. The author listed the various testing types like: functional testing, stress testing, recovery testing, volume testing, User Acceptance Testing , system testing, Sanity or Smoke testing, load

testing, Usability testing, Exploratory testing, ad-hoc testing, alpha testing, beta testing etc.

- **White box testing**

25. Mirza Mahmodd Baig, Burnstein I (2009) has published done research on topic, "**New Software testing strategy**"[25]

Myers G. J. and Burnstein I. present the White box testing strategy also called as structural, glass, and clear box testing strategy. The authors said that the strategy is based on internal logic and structure of the code. In White box testing integrating coverage of the code and also have knowledge of internal working of the code. The author also gave the advantages of the white box strategy i.e. it is easy to find out which type of input /data can help in testing, effectively, optimizing the code, and removing the spare lines of code. The main disadvantage of this testing strategy is that, it increases the cost and time and also impossible to find out hidden errors.

- **Grey box testing [Annexure-II]**

26. Mohd. Ehmer Khan , Farmeena Khan have published research article on "**A Comparative Study of White Box, Black Box and Grey Box Testing Techniques**"[26]

In this research article**, Mohd. Ehmer Khan , Farmeena Khan** have defined grey-box testing is as it is a technique to test the application with having a limited knowledge of the internal workings of an application. [26]

- **Functional testing [Annexure-II]**

27. Paul C. Jorgensen (2013) published book on "**Software testing: a craftsman's approach**" CRC Press[27]

Paul C. Jorgensen present functional tests, or black box testing. Functional testing is a quality assurance process used to verify that an application's end user functionality, i.e. ability to log in and complete transaction etc. He describes that functionality works accurately, reliably, predictably and securely. Functional testing engages either manual or automated testing methods. The author said that manual testing is boring and time

consuming process as compared to automated testing. Its efficiency accelerates the testing cycle and promotes software quality. Automated functional testing can optimizes software quality and efficiency by verifying the accuracy and reliability of an application's end user functionality in pre-production. Functional testing which ensure that simple or even complex enterprise applications are deployed on time and on cost. Today, the main interest of testers and companies around the world is achieving target on time and on cost.

Functional Testing covers: [27]

- The determination of the functionality that the intended application is meant to perform.
- The creation of test data based on the specifications of the application.
- The output based on the test data and the specifications of the application.
- The writing of test scenarios and the execution of test cases.
- The comparison of actual and expected results based on the executed test cases.

- **Non-Functional testing**

28. Wasif Afzal et al. (2008) have published research article on "A Systematic Mapping Study on Non-Functional Search-based Software Testing" [28]

Wasif Afzal et al described about the non-functional software testing. This section is based upon testing an application from its non-functional attributes. Non-functional testing involves testing a software from the requirements which are non-functional in nature but important such as performance, security, user interface, etc.[28]

- **Integration testing**

29. W. K. Chan et al (2002), have published article on "**An Overview of Integration Testing Techniques for Object-Oriented Programs**" [29]

W. K. Chan defined Integration testing as it is the testing of combined parts of an application to determine if they function correctly. Integration testing can be done in two ways: Bottom-up integration testing and Top-down integration testing. [29]

- Bottom-up integration
- Top-down integration

- **System testing**

30. Shivkumar Hasmukhrai Trivedi, (2012), has published research article on "**Software Testing Techniques**"[30]

Shivkumar Hasmukhrai Trivedi stated that system testing tests the system as a whole. Once all the components are integrated, the application as a whole is tested rigorously to see that it meets the specified Quality Standards. This type of testing is performed by a specialized testing team.[30]

- **Regression Testing [Annexure-II]**

31. Leung, H.K.N (1989) has published article on "**Insights into regression testing**"

Leung has mentioned in his research article that whenever a change in a software application is made, it is quite possible that other areas within the application have been affected by this change. Regression testing is performed to verify that a fixed bug hasn't resulted in another functionality or business rule violation. The intent of regression testing is to ensure that a change, such as a bug fix should not result in another fault being uncovered in the application.[31]

- **Acceptance Testing**

32. M. Fagan, has published research article on "**Design and Code Inspections to Reduce Errors in Program Development**"

M. Fagan provided detail about how acceptance testing is arguably the most important type of testing, as it is conducted by the Quality Assurance Team who will gauge whether the application meets the intended specifications and satisfies the client's requirement. The QA team will have a set of pre-written scenarios and test cases that will be used to test the application.[32]

- **Alpha testing**

33. Hitesh Tahbildar at. el. (2011). "**Automated software test data generation: Direction of research**"

Hitesh Tahbildar defines alpha testing as it is the first stage of testing and will be performed amongst the teams (developer and QA teams). Unit testing, integration testing and system testing when combined together is known as alpha testing.[33]

- **Beta Testing**

34. Rishabh Softwares (2011) posted one article on "The Importance of Beta Software Testing in QA"

Rishabh softwares posted research articles about beta testing and they have mentioned about beta testing is: it is performed after alpha testing has been successfully performed. In beta testing, a sample of the intended audience tests the application. Beta testing is also known as pre-release testing.[34]

- **Performance Testing**

35. **Ms. S. Sharmila (2014)**, has published research article on "Analysis of Performance Testing on Web Applications"

Ms. S. Sharmila has defined performance testing as it is mostly used to identify any bottlenecks or performance issues rather than finding bugs in a software. It is mostly used to identify any bottlenecks or performance issues rather than finding bugs in a software.[35]

- **Load Testing**

36. **Pooja Ahlawat** (2013) has published research article on "**A Comparative Analysis of Load Testing Tools Using Optimal Response Rate**"

Pooja Ahlawat has defined load testing as it is a process of testing the behavior of a software by applying maximum load in terms of software accessing and manipulating

large input data. It can be done at both normal and peak load conditions. This type of testing identifies the maximum capacity of software and its behavior at peak time.[36]

3.11. Impact of Poor requirement gathering process on Software Testing Process

37. **Josef H. (2001)**, has written chapter 4 in the book named as '**Capturing the Requirements'.**

Josef has mentioned that quality of requirements can have a lot of impact on the outcome of the project. One high profile project which was significantly affected by the requirements management process was the Chrysler Comprehensive Compensation System which was supposed to handle paychecks for Chrysler's 87,000 employees but was shut down after several years of development [37].

The impact is magnified as the BA moves from high-level requirements towards functional and non-functional requirements. The cost of rework of functional requirements is the highest because these requirements define the technical specification and design of the solution [37]

38. Md Rounok Salehin has published article on "**Missing Requirements Information and its Impact on Software Architectures:A Case Study**"

Md Rounok Salehin has provided the detail literature review for some past project failures mentioned from existing literature which shows the severity of the problems caused by poor (missing or incomplete) requirements in industries.

He has also studied and found a striking 74% project failure rate, while 28% of projects were cancelled completely and top reasons of failure was lack of user input, lack of a clear statement of requirements in specifications.

He has taken Siemens project as the case study and the root cause analysis done by Siemens Corporate Research (SCR) showed that 40% of the defects were caused by incomplete or not at all recorded requirements in documents.

39. **Mohd. Ehmer Khan**, has published article on “Different Forms of Software Testing Techniques for Finding Errors,”

Mohd. Ehmer Khan provided detail study about survey was conducted in 63 software companies in Malaysia. The companies cited problems like incomplete requirements (79.4%), misplaced requirements in a requirements document (37.1%) as some of the reasons behind late delivery of products (76.2%), budget overruns (58.7%) and poor quality products (44.4%).

In June 1991 to September 1991, three surveys on 39, 41 and 44 software maintenance professionals (with overlapping participants) were conducted and 19 major problems in software maintenance were identified [17]. Incomplete information in system documentation was ranked number 3 amongst them.

As per the report [59], industry data suggests that approximately 50% of product defects originate in the requirements. Perhaps 80% of the rework effort on a development project can be traced to requirements defects. These defects are the cause of over 40% of accidents involving safety critical systems.

40. **Christel, Michael and Kyo C. Kang** (September 1992). "Issues in Requirements Elicitation".

Christel, Michael and Kyo C. Kang mentioned the issues in requirement elicitation like “Lack of mechanism of validation and verification” and as per them this is one of the major cause of software project failure. Because system development and testing is probably the most critical phase of any software development project. Adequate programming and testing methods and techniques need to be adopted. The use of unstable and sometimes incompatible software and hardware platforms may pose significant risk to the project. Tools needed for testing and verifying the application or product are indispensable for a successful implementation and deployment of the software. If correct tools are not available on time, it may delay the deployment, thus affecting the overall project schedule [40].

Christel, Michael and Kyo C. Kang also stated that poor requirement happens due to the problems that indicate the challenges for requirements gathering [15].

Following are few challenges that we need to consider while doing requirement gathering.

- '**Problems of scope**'. The boundary of the system is ill-defined or the customers/users specify unnecessary technical detail that may confuse, rather than clarify, overall system objectives.
- **Problems of understanding**. The customers/users are not completely sure of what is needed, have a poor understanding of the capabilities and limitations of their computing environment, don't have a full understanding of the problem domain, have trouble communicating needs to the system engineer, omit information that is believed to be "**obvious**," specify requirements that conflict with the needs of other customers/users, or specify requirements that are ambiguous or untestable.
- **Problems of volatility**. The requirements change over time. The rate of change is sometimes referred to as the level of requirement volatility

41. Donald Firesmith, has published research article on "**Common Requirements Problems, Their Negative Consequences, and the Industry Best Practices to Help Solve Them**". [41]

Donald Firesmith, focused on impact of poor requirement gathering process on software testing. In his research article his mentioned that requirement collection from client is one of the basic and important task of requirement gathering process. He also explained how inadequate information from client lead to low performance of software project. Due to inadequate knowledge of business or functional requirement, business analyst can collect incorrect requirements from client. Inadequate information collected from client definitely going to impact of software development and software testing process. There are many software product failures examples in the world because of incorrect, incomplete requirements. Literature review has shown the many reasons for IT project failure in all over the world. Out of 100% project success rates were only 34% with the rest of project being either "challenged" in some way or failing outright. The failure in software project means there is loss in productivity, revineo of Software

Company and these losses are very significant. For example, British food retailer Sainsbury had to write off its $526 million investment in an automated supply-chain management system. The U.S. Federal Aviation Administration spent $2.6 billion unsuccessfully trying to upgrade its air traffic control system in the 1990s. Ford Motor Company abandoned its purchasing system in 2004, after spending $400 million. In the 8 years since, things probably haven't changed much.

Donald Firesmith provided few causes that leads to poor requirement gathering [41]:

1. Poor Requirements Quality
2. Over Emphasis on Simplistic Use Case Modeling
3. Inappropriate Constraints
4. Requirements Not Traced
5. Excessive Requirements Volatility including Unmanaged Scope Creep
6. Inadequate Verification of Requirements Quality
7. Inadequate Requirements Validation
8. Inadequate Requirements Management
9. Inadequate Requirements Process
10. Inadequate Tool Support
11. Unprepared Requirements Engineers

42. **Indika Perera**, has published research article on "**Impact of Poor Requirement Engineering in Software Outsourcing: A Study on Software Developers' Experience**"

Indika Perera also thrown light on how poor requirement gathering process impacts complete life cycle of software project. She has mentioned main reason of such project failure is incomplete software requirement which in turn happened due to poor requirement gathering. In SDLC process, most of time it is impossible to have complete and finalized set of requirements at the beginning of a project. This leads requirement

changes to happen during the latter stages of the project and create conflicts with the software process been practiced [13].

43. **Ramos Rowel and Kurts Alfeche** (1997) has published article on "**Requirements Engineering A good practice guide**", John Wiley and Sons, 1997

Ramos Rowel and Kurts Alfeche defined requirement gathering as it is the practice of collecting the requirements of a system from users, customers and other stakeholders. [14] Requirements gathering practices include interviews, questionnaires, user observation, workshops, brainstorming, use cases, role playing and prototyping.

One of the root causes of poor requirement gathering in SDLC is the only role for users is in specifying requirements, and that all requirements can be specified in advance. Unfortunately, requirements grow and change throughout the process and beyond, calling for considerable feedback and iterative consultation. Due to this frequently changing requirements gathering process, many developers complain about inadequate, non-freezing requirements and its impact on their work, software productivity and time consuming overhead. This non-freezable requirement gathering process not only affects developer's work but also affecting Tester, maintenance and management team. Non-Freezable requirement leads to poor software requirement gathering and in turn leads to non-qualitative software product. Poor requirement gathering mostly happens due to business problem, and not a technology problem.

The non-freezable requirements for software, as delivered by typical business analysts, designer is not sufficiently clear, insightful, or well understood to develop software systems that meet the needs of business users.

To overcome this problem, there is need to understand root cause of poor software requirements gathering process and find out the corrective solution for the same.

44. Rahul Thakur and Subhajit Dasgupta have published research article on "impact of software requirement volatility pattern on project dynamics: evidences from a case study" International Journal of Software Engineering & Applications (IJSEA), Vol.2, No.3, July 2011

In this paper authors have investigated the impact of requirement volatility pattern on project performance. Various types of efforts are considered through case study, in which authors have shown that effect of volatility of requirements has impact on project, because employees have to put extra efforts for redo task of software development process.

45. "Best Practices for Change Impact Analysis" article on Impact Analysis for Requirement change by Karl Wiegers at Jama Software's on February 19, 2014.

In this article author has explained the concept of requirement change and if change is given by client, how impact analysis technique can be used. In this article he has explained the format of recording change in the document termed as "Proposed Change" and technique termed as Impact Analysis discussed and then total efforts calculated based on proposed change document.

This chapter provides detail literature review about the software, software engineering process, and different types of software development life cycle models. It also gives detail literature review on software testing process and type of software testing. Literature review on Impact of poor requirement gathering process on software testing process is also given in detail.

1. Muhammad Naeem Ahmed Khan and et.all (2013), "Review of Requirements Management Issues in Software Development" I.J.Modern Education and Computer Science, 2013, 1, 21-27, Published Online January 2013 in MECS (http://www.mecs-press.org/) DOI: 10.5815/ijmecs.2013.01.03
2. https://web.cs.dal.ca/~hawkey/3130/SEBackground4.pdf
3. Systems Development Lifecycle: Objectives and Requirements. Bender RPT Inc. 2003

4. Vanshika Rastogi (2015), “Software Development Life Cycle Models- Comparison, Consequences” IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 6 (1) , 2015, 168-172
5. Software Development Life Cycle (SDLC) Yogi Berra presentation
6. Ms. Shikha maheshwari and Prof.Dinesh Ch. Jain 2012 “A Comparative Analysis of Different types of Models in Software Development Life Cycle” International Journal of Advanced Research in Computer Science and Software Engineering, Volume 2, Issue 5, May 2012
7. Royce, Winston (1970), "Managing the Development of Large Software Systems" (PDF), Proceedings of IEEE WESCON 26 (August): 1–9
8. PK.Ragunath, S.Velmourougan, P. Davachelvan, ,S.Kayalvizhi, R.Ravimohan (2010) “Evolving A New Model (SDLC Model-2010) For Software Development Life Cycle (SDLC)” IJCSNS International Journal of Computer Science and Network Security, VOL.10 No.1, January 2010
9. Seema, Sona Malhotra 2012 “Analysis and tabular comparision of popular SDLC models” International Journal of Advances in computing and Information Technology.
10. Sonali MAthur and Shaily Malik (2010), “Advancements in the V-Models”, International Journal of Computer Applications (0975-8887) Volume 1- No.12
11. Naresh Kumar, A. S. Zadgaonkar, Abhinav Shukla “Evolving a New Software Development Life Cycle Model SDLC-2013 with Client Satisfaction” International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-3, Issue-1, March 2013
12. G. Kotonya and I. Sommerville, (1998), have published article on “Requirements Engineering: Processes and Techniques”, in the book published by Chichester, UK: John Wiley & Sons.
13. Requirements Engineering A good practice guide, Ramos Rowel and Kurts Alfeche, John Wiley and Sons, 1997
14. I. Sommerville and P. Sawyer (1997), Requirements Engineering: A Good Practice Guide, New York: John Wiley & Sons,.

15. http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=948567&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D948567
16. K. E. Wiegers, Software Requirements, 2nd ed., Redmond, W A: Microsoft Press, 2003.
17. http://prr.hec.gov.pk/Chapters/369S-2.pdf (Software Testing reference)
18. Donald Firesmith, Software Engineering Institute, U.S.A "Common Requirements Problems, Their Negative Consequences, and the Industry Best Practices to Help Solve Them". http://www.jot.fm/issues/issue_2007_01/column2/
19. Olga Liskin , et al., "Supporting Acceptance Testing in Distributed Software Projects with Integrated Feedback Systems: Experiences and Requirements" 2012 IEEE Seventh International Conference on Global Software Engineering
20. Vishawjyoti, Sachin Sharma, "Study and Analysis of automation testing techniques" , Journal of global research in computer science, Volume 3, No. 12, December 2012, ISSN-2229-371
21. Antonia Bertolino has published his research article on "Software testing research and practice"
22. Vivek Kumar (2012) has published his article on "Comparison of Manual and automation testing" International Journal of Research in Science And Technology, (IJRST) 2012, Vol. No. 1, Issue No. V, Apr-Jun, ISSN: 2249-0604
23. R. M. Sharma (2014), "Quantitative Analysis of Automation and Manual Testing" International Journal of Engineering and Innovative Technology (IJEIT) Volume 4, Issue 1, July 2014
24. Harsh Bhasin, at.el. (2014), have published research article on "Black Box Testing based on Requirement Analysis and Design Specifications"
25. MIRZA MAHMOOD BAIG (2009), "NEW SOFTWARE TESTING STRATEGY" N.E.D. University of Engineering & Technology
26. Mohd. Ehmer Khan , Farmeena Khan "A Comparative Study of White Box, Black Box and Grey Box Testing Techniques", (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 3, No.6, 2012

27. Paul C. Jorgensen (2013) published book on "Software testing: a craftsman's approach" CRC Press

28. Wasif Afzal et al. (2008) "A Systematic Mapping Study on Non-Functional Search-based Software Testing" paper available at http://www.researchgate.net/publication/221391274

29. W. K. Chan et al (2002), have published article on "An Overview of Integration Testing Techniques for Object-Oriented Programs" Proceedings of the 2nd ACIS Annual International Conference on Computer and Information Science (ICIS 2002), International Association for Computer and Information Science, Mt. Pleasant, Michigan (2002)

30. Shivkumar Hasmukhrai Trivedi, (2012), has published research article on "Software Testing Techniques" International Journal of Advanced Research in Computer Science and Software Engineering, Volume 2, Issue 10, October 2012

31. Leung, H.K.N (1989) has published article on "Insights into regression testing" Software Maintenance, 1989., Proceedings., Conference on

32. M. Fagan, "Design and Code Inspections to Reduce Errors in Program Development," IBM Systems Journal, vol. 38, no. 2/3, pp. 258–287, 1999.

33. Hitesh Tahbildar at el(2011). "Automated software test data generation: Direction of research" International Journal of Computer Science & Engineering Survey (IJCSES) Vol.2, No.1, Feb 2011

34. http://www.rishabhsoft.com/blog/beta-testing-the-importance (2011)

35. Ms. S. Sharmila, "Analysis of Performance Testing on Web Applications" International Journal of Advanced Research in Computer and Communication Engineering Vol. 3, Issue 3, March 2014

36. Pooja Ahlawat (2013) "A Comparative Analysis of Load Testing Tools Using Optimal Response Rate" International Journal of Advanced Research in Computer Science and Software Engineering. Volume 3, Issue 5, May 2013

37. Chapter 4 'Capturing the Requirements', http://www.cse.msu.edu/~chengb/RE-491/Papers/atlee-chapter4.pdf

38. 2013, Md Rounok Salehin "Missing Requirements Information and its Impact on Software Architectures:A Case Study" The School of Graduate and Postdoctoral Studies The University of Western Ontario,London, Ontario, Canada
39. Mohd. Ehmer Khan, "Different Forms of Software Testing Techniques for Finding Errors," IJCSI, Vol. 7, Issue 3, No 1, pp 11-16, May 2010
40. Christel, Michael and Kyo C. Kang (September 1992). "Issues in Requirements Elicitation". Technical Report CMU/SEI-92-TR-012. CMU / SEI. Retrieved January 14, 2012.

41. Donald Firesmith, Software Engineering Institute, U.S.A "Common Requirements Problems, Their Negative Consequences, and the Industry Best Practices to Help Solve Them". http://www.jot.fm/issues/issue_2007_01/column2/

42. Indika Perera, "Impact of Poor Requirement Engineering in Software Outsourcing: A Study on Software Developers' Experience". Int. J. of Computers, Communications & Control, ISSN 1841-9836, E-ISSN 1841-9844 Vol. VI (2011), No. 2 (June), pp. 337-348
43. Requirements Engineering A good practice guide, Ramos Rowel and Kurts Alfeche, John Wiley and Sons, 1997
44. "Impact of software requirement volatility pattern on project dynamics: evidences from a case study" International Journal of Software Engineering & Applications (IJSEA), Vol.2, No.3, July 2011
45. http://www.jamasoftware.com/blog/change-impact-analysis-2/

Chapter 4

Data Analysis and Interpretation

4.1 Introduction

In chapter 3 researcher has focused on literature review, in which researcher has given the information about the summarized points about various research journals papers, and designed questions for data collection.

The study is related to the analysis of impact of poor collecting SOP process on software testing. Survey based research methodology has been used to carry out this research.

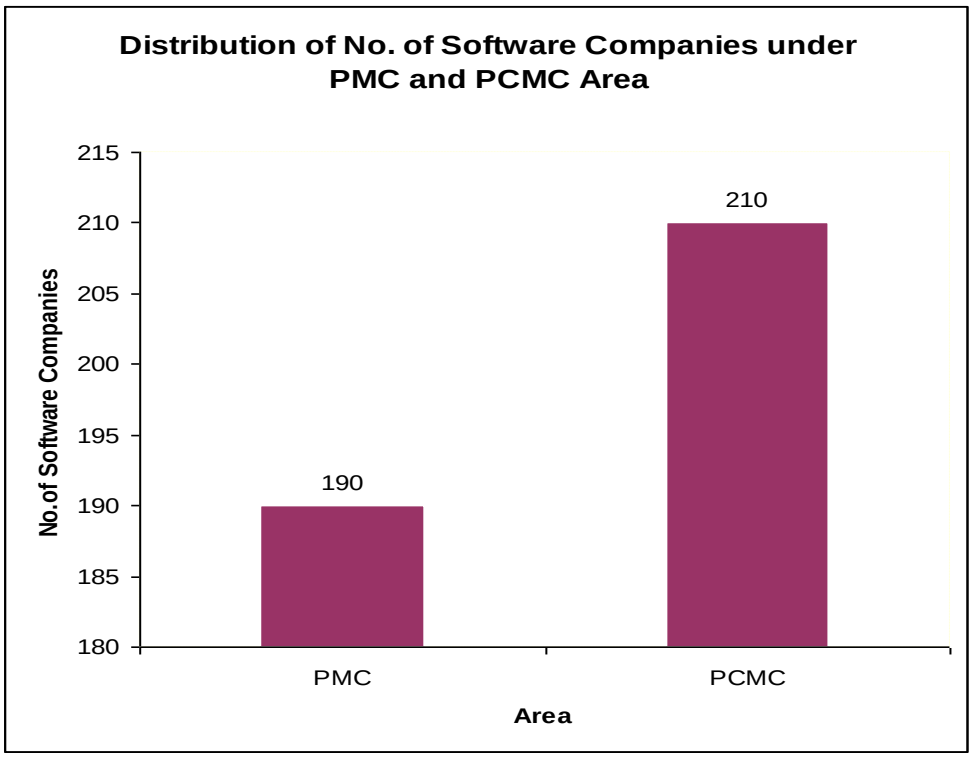
This research is related to the study of impact of collecting SOP on business with special reference in Software companies of Pune city. The researcher has tested positively the hypotheses of this research study, with the help of primary and secondary data.For the purpose of the study, samples have covered all software companies present under PMC and PCMC area. Hence, the researcher has selected one sample viz. software companies present under PMC and PCMC area and collected data from the employees working in these software companies.

4.1.1 Distribution of employees from different software companies present under PMC and PCMC area

Sr. No.	Office Location	Number of Employees form different Software Companies in Pune City
1	PMC	190(47.5%)
2	PCMC	210(52.5%)

Table 4.1. Employees from Software companies present in PMC and PCMC

Above table gives information about employees from PMC and PCMC area of pune.



Graph 4.1: Software companies present in PMC and PCMC

As discussed above, 400 samples are collected from different software companies resids in PMC and PCMC area. Among 400 samples, 190 employees i.e 47.5 % employees of sample belongs to software companies resides in PMC area and 210 employees i.e 52.5% employees of sample belongs to software companies resideds in PCMC area.

The primary data about 400 employees from 21 software companies of Pune city has been collected by the researcher. An analysis is carried out in six broad headings as follows. This analysis is mainly done in the point of view of Business Analyst, Designer and Tester.

4.2 Data Analysis

The data for employees of Software companies is collected through interviews & questionnaires then compiled in 32 tables. Statistical parameters and graphics have been used wherever necessary and useful. The data analysis of this data is as follows:-

1. General Background of Respondents

- Designer and Tester of different software companies with respect to their Gender, Age, Education, Occupation and office location in pune city.

2. Current state of collecting SOP from customer process in Software industry

- From whom and how requirement analyst or designer collecting SOP. And which kind of SOP they are collecting in how much duration?
- Involvement of different people in collecting SOP process and interaction with end user while doing collecting SOP.

- Useful requirement documents for business analyst and requirement engineers.
- Beneficial collecting SOP techniques and significance of requirement documents.
- Business Analyst's time consumption on non-collecting SOP activities.

3. **To study the impact of SOP which are not freezed on business in software development.**
    - Factors responsible for failure of collecting SOP process.
    - Responsible factors which makes software project erroneous
    - Factors responsible for failure of software project
    - Efforts carried out in case of volatility of SOP which has impact on business.
    - Factors affected if client SOP (Statement of Purpose) keep on changing during software development Process.
4. **To analyze various tools used in software companies:**
    - Useful tools for collecting SOP process.
5. **To analyze the current scenario of software testing.**
    - Involvement of testing team throughout the SDLC process.
    - Usage of Testing tools and type of testing like automated or manual testing
    - Number of test cases execution and rate of defect on requirement document.

6. **To understand various hurdles in the software testing and testers problem.**

- Cost involved in testing process in terms of project failure
- effect of poor SOP or requirement on software testing process
- Overheads in software testing
- Common SOP issues responsible to affect software testing process.

4.3. Gender Background of Respondents

In this point general background of respondents like qualification, occupation is discussed.

Employee's basic information like gender and their designation is collected through questionnaire and then it is analyzed in the following table.

4.3.1 Gender and Occupation of Respondents:

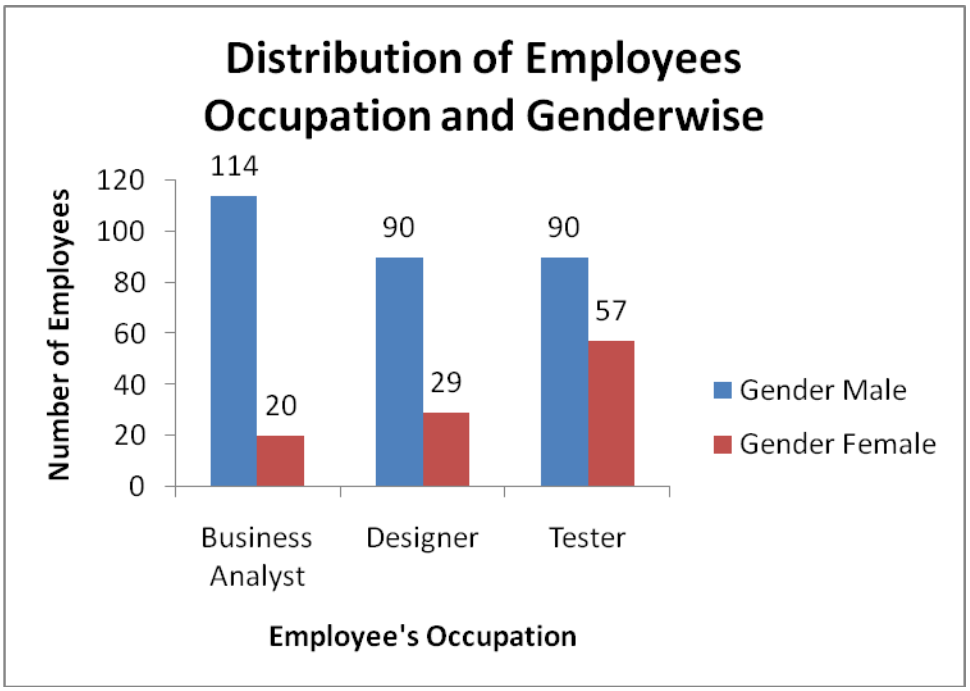
Employee's designation like Business Analyst, Designers and Testers, Business Analyst is a term used for person in Software Company who analyses an organization or business domain and documents its processes, assessing the business model or its integration with technology.

Designer is people who develop the framework for software and interfaces for a system. Tester is a technician who conducts tests on software programs and applications prior to their implementation to ensure quality, design integrity and proper functionality.

For the understanding and awareness about collecting SOP process, researcher has collected educational wise data from 400 employees of different software companies. For this research, this analysis has been done based on gender of employees. From the analysis, it has been seen that 28.5 percent male employees and 5 percent female employees have designation as business analyst and 22.5 percent male and 7.5 percent female employees having designation as Designer.

Occupation	Gender		Total
	Male	Female	
Business Analyst	114 (28.5)	20 (5)	134 (33.5)
Designer	90 (22.5)	29 (7.25)	119 (29.75)
Tester	90 (22.5)	57 (14.25)	147 (36.75)
Total	294	106	400

Table 4.2: Gender and Occupation wise Distribution of Employees



Graph 4.2: Gender and Occupation wise Distribution of Employees

Researcher also collected data for 22.5 percent male and 14.5 percent females employees are working as a Tester. From this analysis, it has been seen that employees are working as Business Analyst, Designer, and tester in different software companies.

4.3.2 Qualification and Occupation wise Distribution of Employees

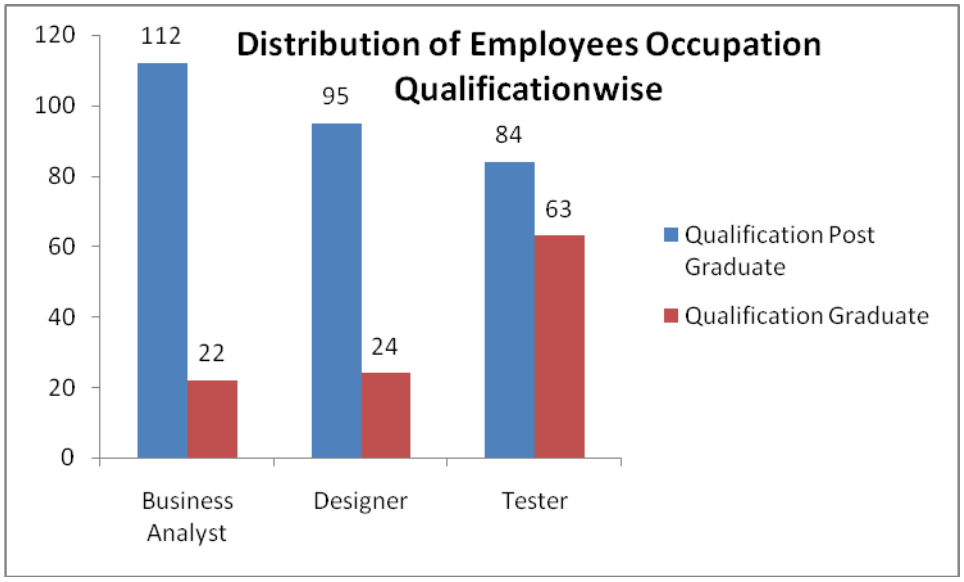
This table gives information about the employee's qualification, like postgraduate and graduate for software development. Here software engineer's designation is as Business Analyst, Designer and Tester.

Business Analyst are those who works as Project Manager who analyze all customer's SOP and prepare Customer requirement Documents.

Designer are those who prepares Logical Documents in which requirement specifications are designed and Testers are those who prepares Test Cases for testing code developed by software developers.

Occupation	Qualification		Total
	Post Graduate	Graduate	
Business Analyst	112 (28)	22 (5.5)	134 (33.5)
Designer	95 (23.75)	24 (6)	119 (29.75)
Tester	84 (21)	63 (15.75)	147 (36.75)
Total	294	106	400

Table 4.3: Qualification and Occupation wise Distribution of Employees



Graph 4.3: Qualification and Occupation Distribution of Employees

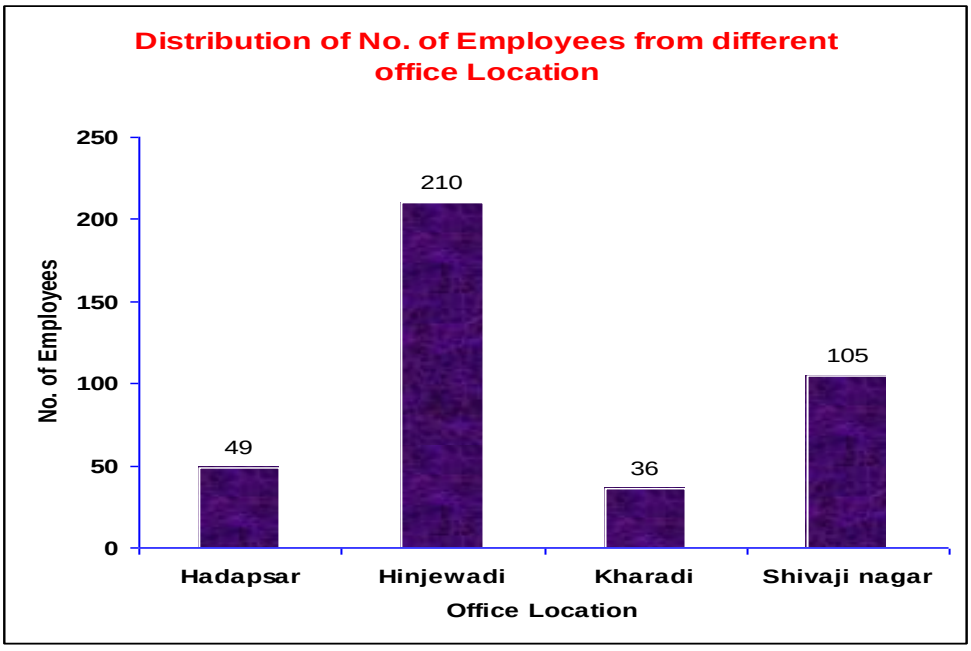
For the understanding and awareness about collecting SOP process, researcher has collected educational wise data from 21 of different software companies. For this research, this analysis has been done based on qualification of employees. From the analysis, it has been seen that 28 percent Postgraduate and 5.5 percent graduate employees are Business Analyst. 23.75 percent postgraduate and 6 percent graduate employees is Designer and 21 percent postgraduate and 15.75 percent graduate employees are Testers. From this analysis, it has been seen that employees are graduate, postgraduate qualified, and they are having knowledge about collecting SOP process with designation wise experience.

4.3.3. Distribution of Employees from different software companies present under PMC and PCMC area

Researcher has collected data from various software companies located in and around pune. Mainly from Hinjewadi Infotech Park and Hadapsar IT park located nearby pune has maximum software companies.

Sr. No.	Office Location	Number of Employees from different Software Companies in Pune City
1	Hadapsar	49 (12.25%)
2	Hinjewadi	210(52.5%)
3	Kharadi	36(9%)
4	Shivaji Nagar	105(26.25%)

Table 4.4: Employees from different software companies present in different areas of pune city



Graph 4.4: Employees from different software companies present in different areas of pune city

For this research, 400 employees are responded from different software companies. These employees are belongs to different software companies with different office locations. As we have considered PMC and PCMC areas from Pune city, Hadapsar, Kharadi and Shivaji nagar office locations belongs to PMC area and Hinjewadi belongs to PCMC area. Hence from the data analysis, it has been seen that 49 employees are from Software companies which are located in Hadapsar area, 210 employees from Hinjewadi area, 36 from Kharadi and 105 from Shivaji Nagar.

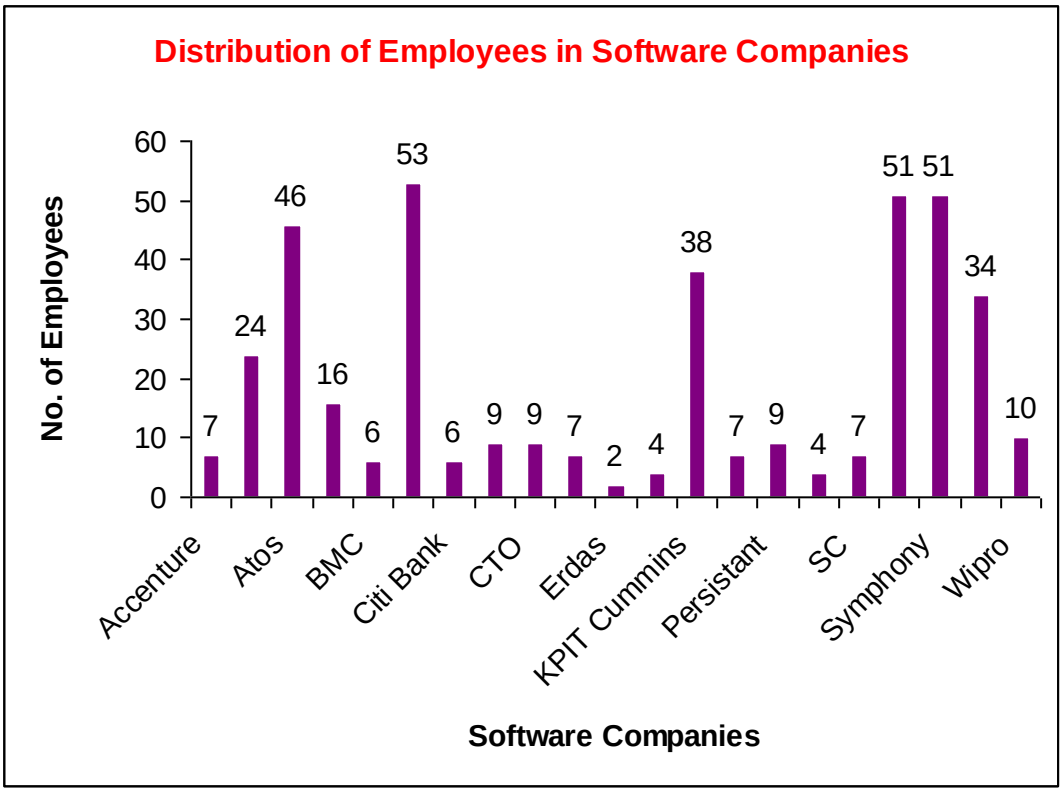
4.3.4. Distribution of Employees in different software companies

In above section we saw, there are four office locations present in pune city and employees from these offices are distributed as per there companies name. Following table shows that number of employees belongs to 21 software companies.

Sr. No.	**Company Name**	**No. of Employees**
1	Accenture	7
2	Amdocs	24
3	Atos	46
4	Davachi	7
5	BMC	6
6	Capegemini	53
7	Citi Bank	6

8	Congnizant	9
9	Hummingbird	9
10	Calsoft	16
11	Neptune Info Tech	2
12	IBM	4
13	KPIT Cummins	38
14	Patani	7
15	Persistent	9
16	Principal Optima	4
17	CLSA	7
18	Sigma Soft	51
19	Symphony	51
20	Tech Mahindra	34
21	Wipro	10

Table 4.5: Distribution of Employees in different software companies



Graph: 4.5: Distribution of Employees in different software companies

4.4 Current state of Collecting SOP Process in Software Industry

4.4.1. Collecting SOP Techniques used in Software Industry:

Customer's SOP analysis, also called customer's SOP engineering, is the process of determining user expectations for a new or modified product. These features, called customer's SOP, must be quantifiable, relevant and detailed. In software engineering, such customer's SOP are often called functional specifications. Customer's SOP analysis is an important aspect of project management.

Customer's SOP analysis involves frequent communication with system users to determine specific feature expectations, resolution of conflict or ambiguity in customer's

SOP as demanded by the various users or groups of users, avoidance of feature creep and documentation of all aspects of the project development process from start to finish.

In today's IT world, more and more software applications and tools are used. If these applications or tools do not work properly according to the customer requirement specifications then it will lead to the failure of software product. To meet the customer requirement specifications there is need to take error free requirement from client. In the current state of software engineering, business analyst and designer use many collecting SOP techniques but for error free customer's SOP, there is need to understand which collecting SOP technique is correct and most suitable.

Hence in this research, survey has been done to identify better and suitable collecting SOP technique. In this survey basically three collecting SOP techniques has been considered:

1. Personally Meeting
2. Through Documents
3. Online-Automated

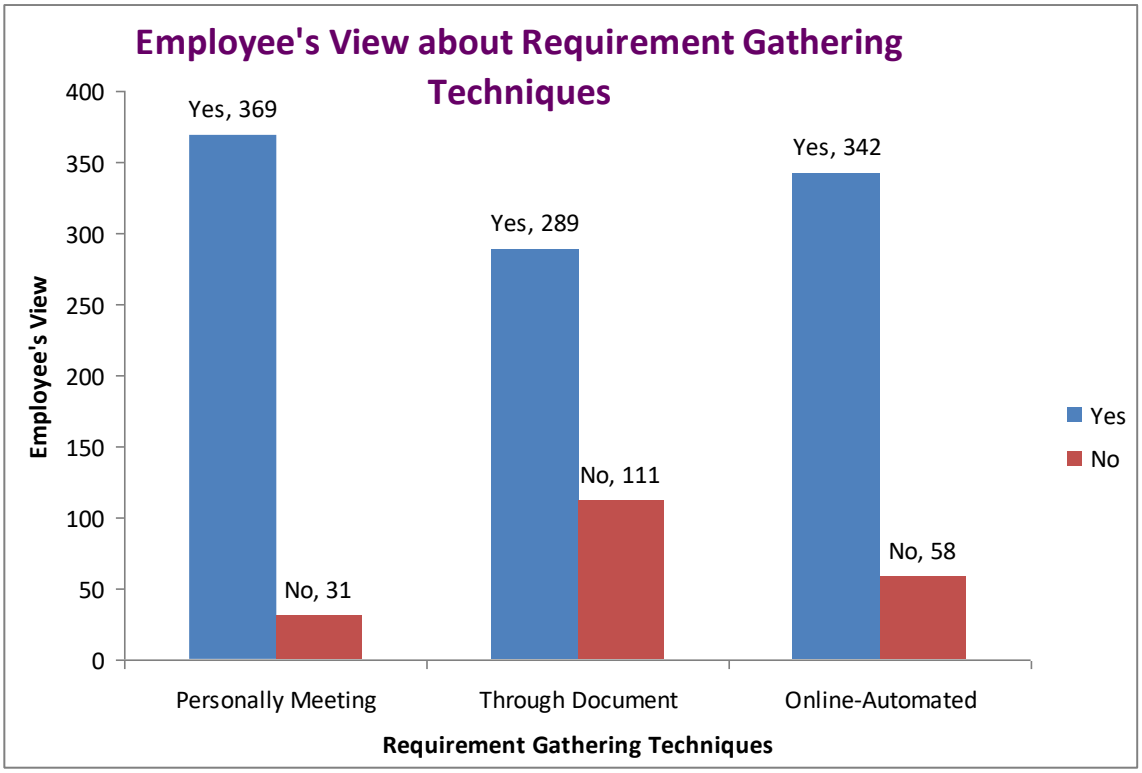
- Personally Meeting is the technique in which client, end user and development team discusses the customer's SOP and prepares requirement document.
- Through Documents is the techniques in which existing documents have been observed like existing reports format, data sheets through customer's SOP can be gathered.
- Online – Automated is a technique through which customer's SOP are uploaded on a shared server, from where customer's SOP can be gathered.

Means using any of these three collecting SOP techniques, business analyst and designer can collect error free requirement from client.

From the analysis of respondent views, it has been shown that, Personally Meeting with client is the best collecting SOP technique as it is nothing but face to face communication with client and using this technique business analyst can easily get clarified all the doubts regarding requirement specification. Hence, maximum around 369 respondents provided positive vote for "Personal Meeting" collecting SOP technique. Some of respondents are also given preference to "Through Document" and "Online-Automated" techniques. As using "Through Document" technique, business analyst can get requirement in written and hence in future there will not be misunderstanding between client and business analyst. Two hundred and Eighty nine people are saying Through Document is a good technique to use for collecting SOP process. Online-Automated technique is useful when shared server is available for client as well as software product provider. Therefore, that in one place all customer's SOP can stored and client as well business analyst can access it at any time. Hence, around 342 respondents agreed for use of Online-Automated tool.

Collecting SOP Techniques	**Yes**	**No**	**Percent**
Personally Meeting	**369**	**31**	**92.3**
Through Document	**289**	**111**	**72.3**
Online-Automated	**342**	**58**	**85.5**

Table 4.6: Collecting SOP Technique used in Software Industry



Graph 4.6: Collecting SOP Technique used in Software Industry

4.4.2. Types of customer's need to use for betterment of software project

As we have seen in chapter 3 that collecting SOP is the process of collection of demands, expectations from the end user or customer to build up a quality product. But in general these customer's SOP should be categorized in high level types like how business analyst or designer can collect requirement from customer. In this research, broad category of requirement types has been defined and they are as follow [2]:

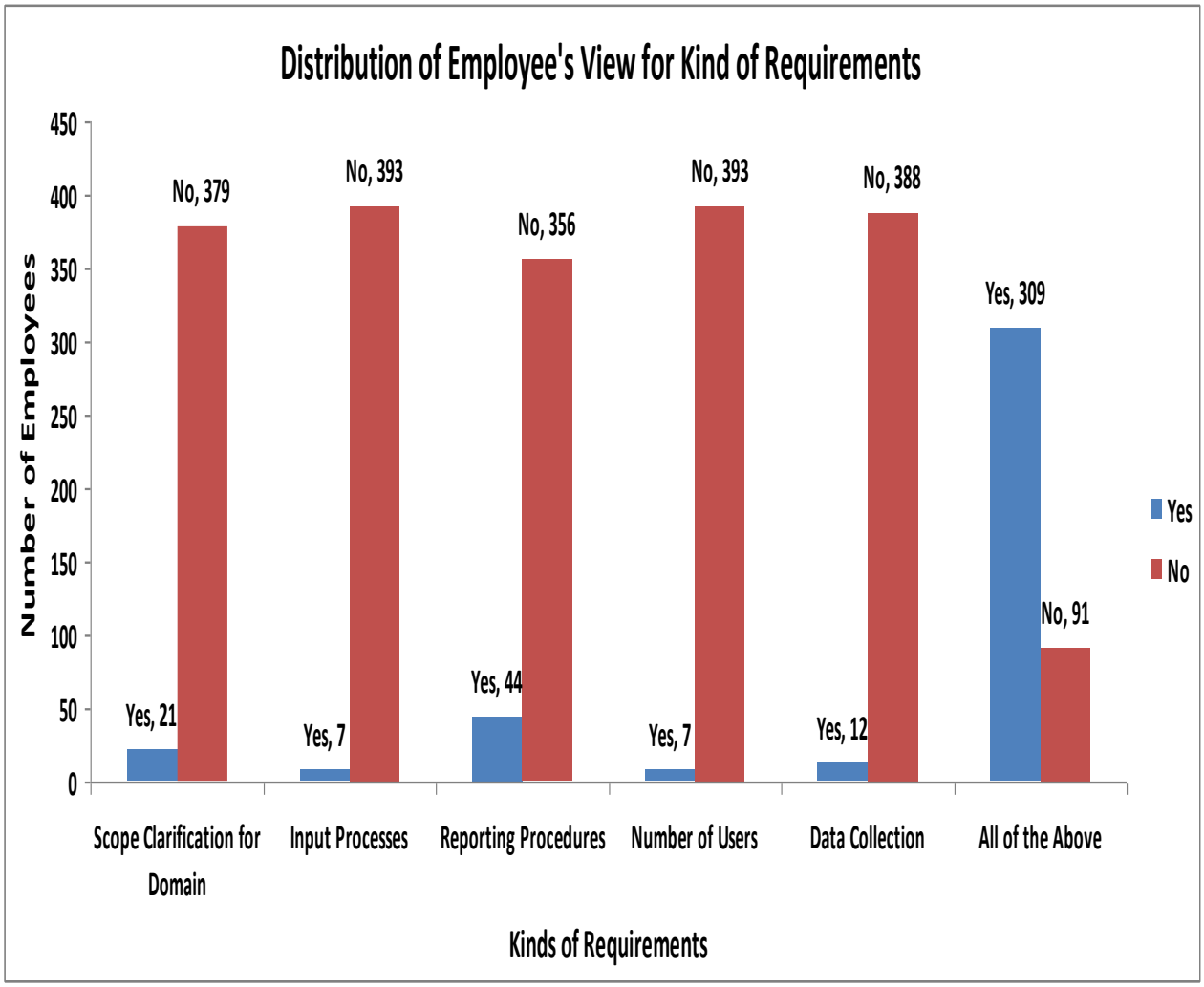
1. Scope Clarification for Domain
2. Input Processes
3. Reporting Procedures
4. Number of Users: Number of users going to be uses proposed software or system.

5. Data Collection : Input data required to be processed and output generated by proposed software or system

In this research, survey has been done on 400 employees of software companies to identify which of the requirement collection method is correct and most suitable for successful software project [See Table 4.7 and Graph 4.7].

Kind of Customer's SOP	**Yes**	**No**	**Percent(Yes)**
Scope Clarification for Domain	21	379	5.3
Input Processes	7	393	1.8
Reporting Procedures	44	356	11.0
Number of Users	7	393	1.8
Data Collection	12	388	3
All of the Above	309	91	77.3

Table 4.7: Types of customer's SOP need to use for betterment of software project



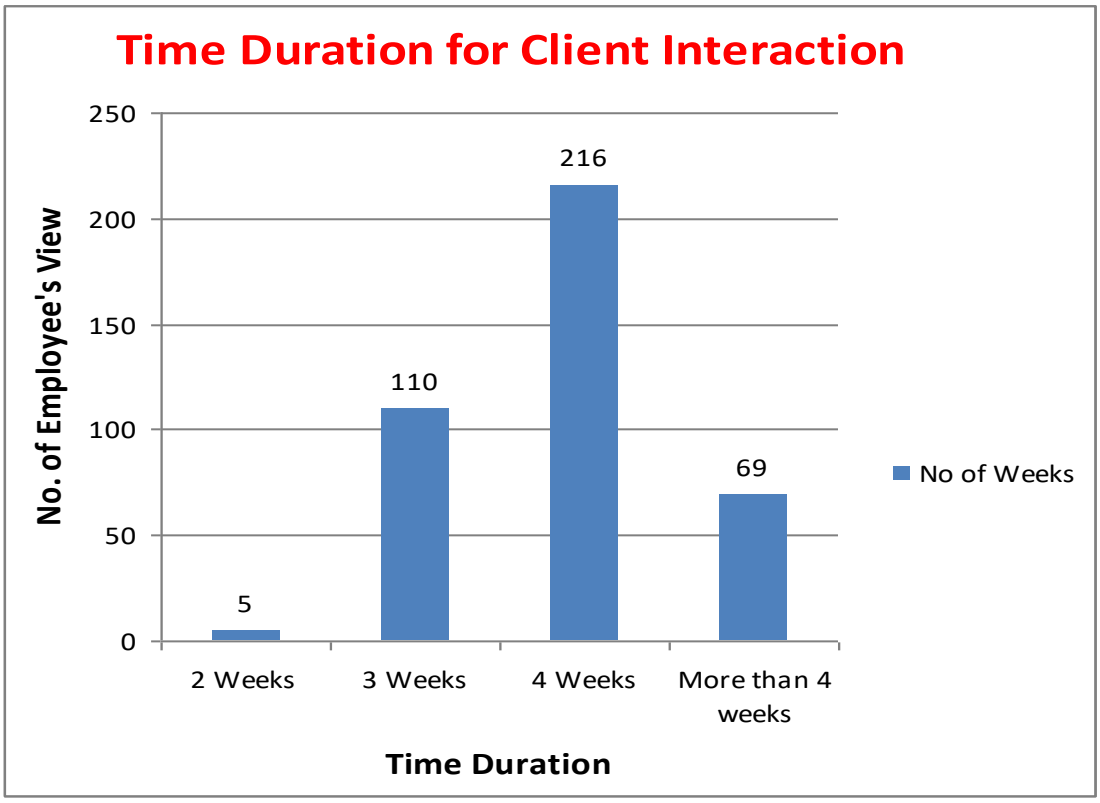
Graph 4.7: Types of customer's SOP need to use for betterment of software project

4.4.3. Time Duration for Client interaction while collecting customer SOP.

Time duration is most important factor and playing vital role in the success or failure of any software project. Time duration required to collect customer's SOP/demands from customer need to be considered in project management process.

No of Weeks	No. of Employee's View	Percentage
2 Weeks	5	2
3 Weeks	110	19.6
4 Weeks	216	50.8
More than 4 weeks	69	27.6
Total	400	100

Table 4.8: Time Duration for Client interaction while gathering customer's SOP



Graph 4.8: Time Duration for Client interaction while gathering customer's SOP

To collect error free requirement from client, there is need to consume adequate time for collecting SOP process. Very less time can lead to incomplete, incorrect customer's SOP but if we consider respondent point of view then in the survey or data analysis we can recognize exact suitable and adequate time for collecting SOP process. Among 400 employees of different software companies, 5 employees are saying 2 weeks are very less time to collect customer's SOP from client. However, for small scale projects, 3 weeks are enough to collect customer's SOP. However, maximum employees are voting to 4 weeks time duration for collecting SOP process. As for large scale project, business analyst SOP to attend many sessions and meeting with client to collect customer's SOP. Hence, 4 weeks time duration is quite adequate to attend sessions or meetings with customer, their system architectures and management team. Around 216 employees are agreed with 4 weeks time duration for requirement collection. Also business analyst need to explain requirement specification in detail to testing and development team, so it also take time to finish actual requirement time and hence sometime requirement phase takes more than four weeks to complete. Hence, 69 employees are saying more than 4 weeks are required to collect customer's SOP from client or customer.

4.4.4. Useful traps for collecting SOP process

In 5.4.1 section, we saw using different collecting SOP techniques like personally meeting, Through Document and Online-Automated we can collect customer's SOP from client. But even though we follow these techniques properly still few mistakes can happen by business analyst or designer and create incorrect, incomplete kind of customer's SOP.

To avoid this incorrect and incomplete requirement issue, there is need to user few traps, these traps are actually suggested by IBM and it SOP to take in practice of collecting SOP process. Following are the traps SOP to be used while gathering customer's SOP from client.

1. **Power up communications with Visuals**: Need to give visual presentation for better visibility of requirement understanding.
2. **Use of standard templates** to support collecting SOP work : Readymade templates with best and standard practices of requirement engineering should be available
3. **Avoid common pitfalls**: Need to avoid common mistakes that are happen frequently while gathering customer's SOP from client.
4. **Need to use Tools** : Need to use automated, online collecting SOP tools to save time of requirement engineering process

Sr. No.	Useful Traps	SA(5)	A(4)	N(3)	D(2)	SD(1)	Avg
1	Power up Communication with visuals	207	136	57	0	0	7.03
2	Use of standard template to support your work	104	231	65	0	0	6.58
3	Avoid common pitfalls	88	255	57	0	0	6.55
4	Uses of tools	86	314	0	0	0	6.77

Table 4.9: Useful traps for collecting SOP process

In this research, 400 respondents responded for their view about useful traps for the betterment of collecting SOP process. 207 employees are strongly recommending for "Power up communication with visuals" because communications with visuals provides more visibility in requirement understanding. 104 employees are strongly agreed for "User of standard template to support collecting SOP work" as standard template is designed after considering best practices for collecting SOP process and hence it is quite useful for error free collecting SOP process. 88 employees strongly recommends "Avoid common pitfalls" means common mistakes SOP to avoid while gathering customer's SOP from client. 86 employees are strongly responded "Use of Tools" option as using automated collecting SOP tools saves more time and lead to increase the productivity of requirement engineering team.

4.4.5. Involvement of different people in collecting SOP process

As we have discuss the various techniques used for collecting SOP and analysis, personally meeting is one of the technique used for requirement collection , in which many users are involved, so here we are discussing which all different users should involve in collecting SOP process.

As we saw 4 week time duration quite fine for requirement collection from client, but along with the time it is quite necessary to involve many people in the collecting SOP process. As Collecting SOP covers many pros and cons like if requirement is error free then there is lots of possibility of success of software project but if anything miss by business analyst then it leads to error in customer's SOP. Only person might be miss few

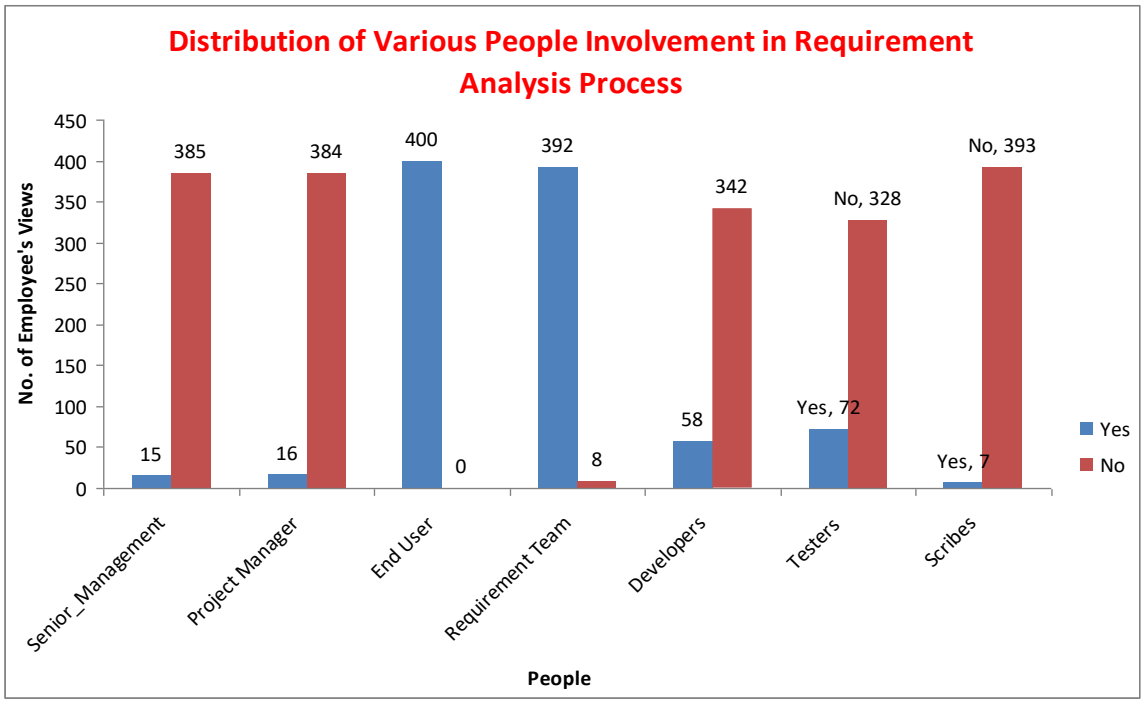
important points or alternation while collecting customer's SOP. There is big possibility of lack knowledge about existing system or product and that lead to incorrect or incomplete requirement collection from client. Hence, instead of only one person like business analyst is not enough for correct customer's SOP collection. There is need to involve many people like Senior management team, senior architecture team, testers, developers, clients, end users and subscribers in the discussion of collecting SOP meeting or session. Because if any person miss any important point or not from the collecting SOP discussion then another might be catch that point and likewise correct and complete requirement can get collected from client. Also due to involvement of senior management, project management, testers and developers all the aspects of system (to be developed) are getting covered and considered.

From the survey, it has been shown that 392 employees are agreed for the involvement of requirement team in the collecting SOP discussion/meeting. It but obvious that requirement team is mandatory team to attend this meeting as these people mainly responsible for requirement collection from client. In addition, 400 employees are responded for the involvement of end user or client as it is also mandatory person as he is actually giving business to Software Company. 58 employees are agreed for the involvement of developers in the collecting SOP discussion. Here developer means technical lead or team lead of software project as he should be capable to provide technical functionality of software product and need to clarify technical issues at that time. Also 72 people are saying tester involvement is mandatory in collecting SOP discussion as testing team verifying and validating the actual business functionality with developed software product. Testing team is responsible for the verification and

validation of software product functionality and should fulfill the customer or end user demands or customer's SOP. Hence testing team is mandatory as a part of collecting SOP discussion.

People involved	**Yes**	**No**
Senior Management	15	385
Project Manager	16	384
End User	400	0
Requirement Team	392	8
Developers	58	342
Testers	72	328
Scribes	7	393

Table 4.10: People Involvement in collecting SOP process



Graph 4.9: People Involvement in collecting SOP process

4.4.6. Time duration required to interact with end user for collecting SOP.

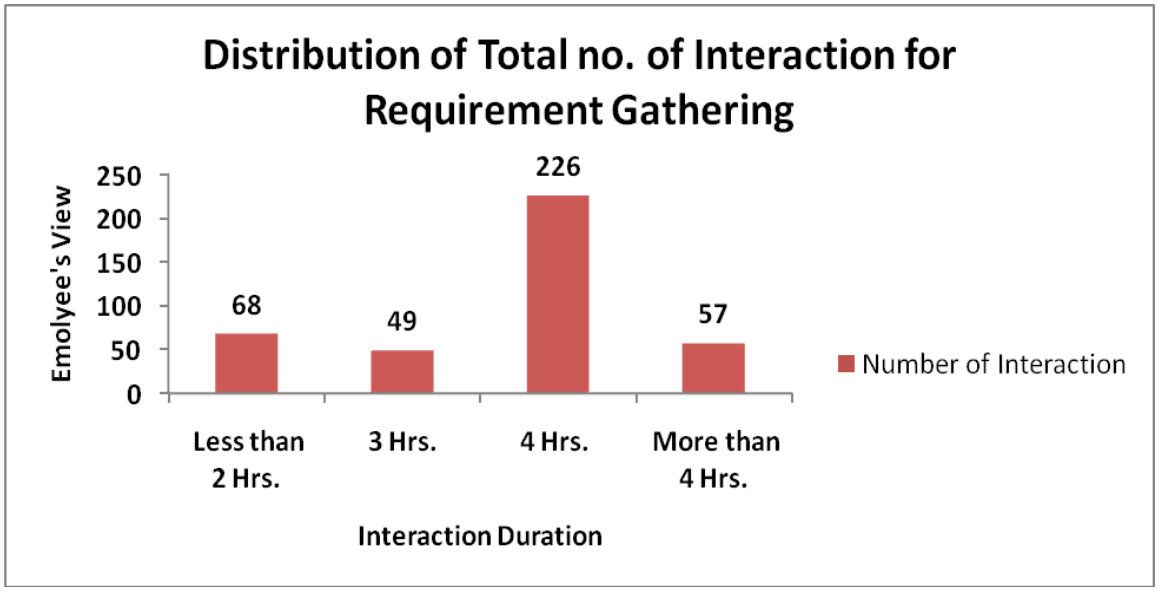
To estimate cost of any software project, time is bigger factor to consider. Cost is always related to time of resources, hardware, software etc. Hence while doing project management, management team need to consider time as important factor. Time SOP in all phases of SDLC. Hence, in collecting SOP phase time required by designer or business analyst to interact with client becomes most important factor. If requirement is even though small but if it complex then business analyst may require more time to get it from end user. If client interaction time gets more as compare to estimated time then it may affect to delivery of software project or quality of project. Hence, there is need to consider how much time designer or business analyst SOP to

interact with client on daily basis. Interaction with client can be done using online call, offline chat, writing mails etc.

In this research, [see table and graph] survey has been done and among 400 employees, 226 employees are strongly recommends daily 4 hrs required to interact with client or end user to understand requirement or to clarify requirement related queries.

Time Duration	Employee's View	Percent
Less than 2 Hrs.	68	17.0
3 Hrs.	49	12.3
4 Hrs.	226	56.5
More than 4 Hrs.	57	14.3

Table 4.11: Time duration required to interact with end user for collecting SOP



Graph 4.10: Time duration required to interact with end user for collecting SOP

Most of time, if discussion is bigger, then more than four hrs is also required and for this point around 57 employees says yes. Sometime requirement discussion topic quite small and hence 68 employees responded for less than 2 hrs option.49 employees are saying 3 hrs are enough to interact with client

1. **The information from Table 4.6(Collecting SOP techniques proves the second objective.), Table 4.7(Types of customer's SOP need to use for betterment of software project), Table 4.8(Time Duration for Client interaction while gathering customer's SOP), Table 4.9(Useful traps for collecting SOP process), Table 4.10(People Involvement in collecting SOP process), Table 4.11(Time duration required to interact with end user for collecting SOP) proves *First Objective "To study various task undertaken for software development process in IT Companies".***

4.4.7. Significance of Different types of Requirement Documents.

In collecting SOP session or meeting, business analyst SOP to take customer's SOP verbally from client and then he/she SOP to record or write requirement in specific document. Recording customer's SOP somewhere in document is a need of an hour because for further changes or future use we need base of requirement. Hence, to record such customer's SOP, software engineering designed and defined different documents like:

1. Customer Requirement Document (CRD)
2. Business Requirement Document (BRD)
3. Functional Requirement Document (FRD)

4. Component Specification Document (CSD)
5. Component Design Document (CDD)
6. Test Case Document (TCD)

It is quite important to understand, among six documents which document is playing most vital role in the success of software project. Table No. 4.12 shows the various documents used in SDLC process. To meet the objective a questionnaire has been designed by using various documents, which define the various documents responsible for success of any software project. It is observed that for each document the average scale is in between 1 to 5 that is in between strongly disagree to strongly agree. In fact all the values are above 3.5 which mean that with respect to all the parameters much approval is observed. In a 5-point Likert scale, having categories like strongly agree, agree, neutral, disagree and strongly disagree clubbed into three categories. The reason for using Likert scale is that the responses by the respondents should not become monotonous while answering the questions. Hence researcher has also applied 5-point Likert scale and calculates weighted average value. There is very less difference between the comparative value of rank order average value and 5-point Likert scale value. [1]

It is seen that the highest average value is 7.71 for the 'Functional Requirement Document (FRD)' followed by 'Component Design Document (CDD)' that is 7.32 and 'Component specification document (CSD)' and 'Test Case Document (TCD)' which are 7.29 and 7.02. The average value for factor 'Customer Requirement Document (CRD)' is 6.78 followed by 'Business Requirement Document (BRD)' is 6.63. It is clear from the average values that Functional Requirement Document (FRD) is most important document as per most of respondents. As FRD is created for the functionality of a

proposed software and it is useful for Testing, development and designer team as well. Hence, most preference has been given to this document. FRD always contains the detail information about integrated functionality of proposed software and customer's demands or requirement in the technical language. Hence, FRD should always be in sync with updated requirement and updated functionality of domain. Component design document (CDD) actually contains the information about technical, business functionality developed component of proposed software, and hence this document is mainly useful for development and testing team. Testing team uses CDD document just to understand business functionality in technical terms. Component Specification Document (CSD) is needed for designer and development team and hence employees from both these teams have given preference to this document. Test Case Document (TCD) is mainly important document for testing team and hence this document should be in sync with FRD that is the business functionality of proposed document. Customer Requirement Document (CRD) is the base for all documents. This document actually created by customer itself and accordingly business analyst and designer creates BRD, FRD, and other documents. CRD always contains customer's specific demands or customer's SOP and it is available in the customer's understanding language. BRD is created by considering CRD and it contains only business functionality of proposed software. It does not have technical functionality and hence this document is only helpful for business analyst and system architecture team.

Sr.No	Factors	SA(5)	A(4)	N(3)	D(2)	SD(1)	Avg
1	Customer Requirement Document (CRD)	160	183	42	15	0	6.78
2	Business Requirement Document (BRD)	162	126	112	0	0	6.63
3	Functional Requirement Document (FRD)	334	51	15	0	0	7.71
4	Component Specification Document (CSD)	223	170	7	0	0	7.29
5	Component Design Document (CDD)	229	164	7	0	0	7.32
6	Test cases Document (TCD)	190	169	41	0	0	7.02

Table 4.12: Significance of Different types of Requirement Documents

4.4.8. Time consumption of Business Analyst on sub activity of collecting SOP from customer.

As we saw, different factors like inadequate resources and inadequate time are responsible for the failure of collecting SOP process. Hence, for time and resources management there is needed to consider work schedule and number of activities performed by collecting SOP team members. In many organizations, it has been seen that due to workload, business analyst need to work on non-collecting SOP activities as well. Hence, due to this extra workload, business analyst could not focus on his/her actual collecting SOP activities, which in turn leads to incorrect, incomplete customer's SOP.

In this research, following factors has been considered as non-requirement activities performed by business analyst and designer.

1. Writing Requirement Documents
2. Reviewing FRD/BRD
3. Client Customer Interaction
4. Conducting Training for Testers and Developers

- Writing Requirement Documents is the process in which collected customer's SOP are stored in the specific format by business analyst and these documents are distributed for next phase of software development.
- Reviewing FRD/BRD in this function requirement document or business required document is verified for checking customer's customer's SOP.
- Client Customer Interaction, in this periodically meetings must be conducted with client or customer for showing progress of development of product and for verifying customer's SOP.

To get views from respondents, data is divided in different ranges of percentage like 0-25, 25-50, 50-75, 75 and above. From the survey, it has been seen that higher percentage range value for 'Writing Requirement Document' is 129.

	Percentage			
Factors	**0-25**	**25-50**	**50-75**	**75 and above**
Writing Requirement Document	138	133	129	0
Reviewing FRD/BRD	265	135	0	0
Client Customer Interaction	15	160	210	15
Conducting Training for Testers and Developers	306	62	32	0

Table 4.13: Time consumption of Business Analyst on non-collecting SOP activities

4.5 To Study the Impact of SOP which are not freeze in Software Development

4.5.1 Failure of Collecting SOP process

Some factors are responsible for failure of collecting SOP process which is discussed below.

- **Lack of knowledge about the business context** means, when initially customer's SOP which gather from client are consider as business context, if developer team has less knowledge about domain, it is considered as lack of knowledge about business context.
- **Lack of understanding of Business** -: Customer SOP or demands are nothing but problems or opportunities and business analyst, designer SOP to find out solutions for those problems. But if business analyst or designer will have lack understanding about the business problems then how he will find out exact

solutions for those problems. Hence Understanding of Business problems/opportunities is most important factor to reduce failures in collecting SOP process.

- **Missing of gaps to be bridged**-: In software development process at various stages or phases software is developed so there must be communication at every stage about development of software so that some missing customer's SOP can be carried out in development process.
- **Inadequate number of Resources** -: is the term used for supportive resources like time, knowledge, tools which used for collecting SOP process.
- **Inadequate Time**-: Here if business analyst spends adequate (satisfactory) time for collecting SOP process then here less chance of gathering wrong or missing customer's SOP.

Strongly Agree (SA)-5, Agree(A)-4, Neutral(N)-3, Disagree(D)-2, Strongly Disagree(DS)-1							
Sr. No.	**Factors**	**SA(5)**	**A(4)**	**N(3)**	**D(2)**	**SD(1)**	**Avg**
a)	Lack of Knowledge about the business context	280	113	7	0	0	7.52
b)	Lack of Understanding of Business problems/opportunities	252	148	0	0	0	7.44
c)	Missing of gaps to be bridged	187	213	0	0	0	7.18
d)	Inadequate number of Resources	116	250	34	0	0	6.76
e)	Inadequate Time	181	185	34	0	0	7.02

Table 4.14: Factors affecting on collecting SOP

A collecting need is the process of collection of demands, expectations from the end user or customer to build up a quality product. Collecting SOP is the basic and important step of requirement engineering stream. The main goal of collecting SOP process to get exact demands SOP from the end user or customer in detail without missing any minor note.

But during collecting SOP process many challenges can encounter like Changing nature of customer's SOP, inadequate communication, problem of scope, incomplete customer's SOP, ambiguous customer's SOP, wrong selection of stake holders, inappropriate selection techniques, conflicting customer's SOP are some of the problems [3]. 90% of large software projects are failing due to poor collecting SOP process [4].

Hence, researchers has designed question, which is focused on different factors responsible for failure of collecting SOP process. Because collecting SOP process playing most vital role in the success or failure of software project.

Table No. 4.14 shows the various factors responsible for failure of collecting SOP process. To meet the objective a questionnaire has been designed by using various factors, which define the various points responsible for success of any software project. It is observed that for each factor the average scale is in between 1 to 5 that is in between strongly disagree to strongly agree. In fact all the values are above 3.5 which mean that with respect to all the parameters much approval is observed. In a 5-point Likert scale, having categories like strongly agree, agree, neutral, disagree and strongly disagree clubbed into three categories. The reason for using Likert scale is that the responses by the respondents should not become monotonous while answering the questions. Hence

researcher has also applied 5-point Likert scale and calculates weighted average value. There is very less difference between the comparative value of rank order average value and 5-point Likert scale value. [1]

It is seen that the highest average value is 7.52 for the 'Lack of knowledge about the business context' is followed by 'Lack of understanding of Business problems/opportunities' is 7.44, followed by 'Missing of gaps to be bridged' that is 7.18, followed by Inadequate Time' is 7.02 and 'Inadequate number of Resources' is 6.76.

It is clear from the average values that 'Inadequate number of Resources' is most responsible for the failure of collecting SOP process. As per most of average like 6.76 of respondents are specifying inadequate number of resources means crunch of business analyst or designer in the requirement team and due to this one business analyst need to do work of 2 or 3 hours and hence it impacts on quality of work in turn quality of customer's SOP. Hence, researcher is strongly recommends that adequate number of Resources is the need of an hour for the success of collecting SOP process and in turn success of any software project.

Along with adequate number of resources, business analyst should have knowledge about business context. Knowledge about business context is the basic stuff for any collecting SOP process. If business analyst does not have knowledge about business context then it may lead to incorrect collecting SOP and will in turn lead to failure in collecting SOP process. Hence, each business analyst or designer should be trained for the knowledge of business context before attending the collecting SOP meeting.

7.44 percentage of respondents are saying 'Lack of understanding of Business problems/opportunities' is the again responsible factor for the failure of collecting SOP

process. Customer SOP or demands are nothing but problems or opportunities and business analyst, designer SOP to find out solutions for those problems. But if business analyst or designer will have lack of understanding about the business problems then how he will find out exact solutions for those problems. Hence Understanding of Business problems/opportunities is most important factor to reduce failures in collecting SOP process.

If business analyst understands problems or opportunity provided by customer then he SOP to understand whether Software Company's product can provide solutions for customer's problems. If his product can not support functionality demanded by customer then he SOP to understand gaps between customer demands and product functionality. If business analysts miss any important point in customer's SOP gathering meetings then it is leading to gap to be bridged. 7.18 respondents agreed that 'Missing of gaps to be bridged' leads to failure in collecting SOP process. Hence, researcher recommends that business analyst or designer should identify gaps that to be bridged throughout SDLC process.

Time is the biggest factor for any work to be successful. Hence, for software project also to be successful there is need to have adequate time to finish work. But most of time, due to project cost overrun, cost cutting is taken place in time and it is always bothering requirement team to finish collecting SOP process within very short period of time and this kind of inadequate time in collecting SOP work lead to incorrect, incomplete and non-qualitative customer's SOP. Such kind of inadequate time factor affects to the quality of software product and in turn leads to failure of software product. Hence, this research recommends that adequate time should be allocated to collecting SOP work to

collect qualitative customer's SOP. As requirement is the base for overall software product quality, hence need to consider qualitative customer's SOP.

2. **Information from Table 4.12 (Significance of Different types of Requirement Documents), Table 4.13 (Time consumption of Business Analyst on non-collecting SOP activities), Table 4.14 (Factors affecting on Collecting SOP) proves third objective *"To identify various factors responsible for the software development".***

4.5.2. Factors responsible to make software erroneous

Software testing is a branch where verification of software's functionality is happening. Once development team complete their development and submit product to testing team, then testing team start verification of functionality of software via test cases execution. However, most of the time software having too much issues or errors because of many factors. Hence, it is quite important to understand which factors are responsible to make software erroneous. In this research following list of factors are considered

1. Logic Design-: 'Logic Design' is one of the factors responsible for erroneous software product. Design of logic generally comes into Component design document and Functional Requirement Document (FRD).
2. Documentation-: Requirement, development and testing mainly relay on document to complete their work. Requirement team relay on Customer Requirement Document (CRD) to complete Functional Requirement Document (FRD), Business Requirement Document (BRD) and Component Specification document (CSD).

3. Human-: Human is responsible factor to make software error as in most of manual processes human makes mistakes and it leads to failure of software.
4. Environment-: Environment is nothing but hardware on which software is getting executed.
5. Data-: 'Data' factor is also responsible to make software erroneous because software testing is getting done by using dummy data
6. Interface-: 'Interface' is nothing but rule or protocol by using that two systems are communicating or sending messages to each other.
7. Requirement Errors-: factor to make software product erroneous because as we know that requirement is the base for all the phases of SDLC process. If base is incorrect or incomplete then obviously it affects the flow of SDLC process. Requirement errors always affect the productivity of development, testing team.

Above listed factors are mostly found as errors in software product. Hence, it is quite important to understand which factor is most responsible to make software product erroneous. Table No. 4.15 shows the various factors responsible for erroneous software. To meet the objective a questionnaire has been designed by using various erroneous factors which makes software product erroneous. It is observed that for each factor has the average scale in between 1 to 5 that is in between strongly disagree to strongly agree. In fact all the values are above 3.5 which mean that with respect to all the parameters much approval is observed. In a 5-point Likert scale, having categories like strongly agree, agree, neutral, disagree and strongly disagree clubbed into three categories. The

reason for using Likert scale is that the responses by the respondents should not become monotonous while answering the questions. Hence researcher has also applied 5-point Likert scale and calculates weighted average value. There is very less difference between the comparative value of rank order average value and 5-point Likert scale value. [1]

It is seen that the highest average value is 4.73 for the 'Requirement Errors' followed by 'Logic Design' is 4.34. The average value of 'Documentation' is 4.26 followed by 'Data' and 'Environment' that is 3.74. The average value of 'Interface' is 3.58 followed by 'Human' that is 3.16.

As per 4.73 percentage of respondents 'Requirement Errors' is most responsible factor to make software product erroneous because as we know that requirement is the base for all the phases of SDLC process. If base is incorrect or incomplete then obviously it affects the flow of SDLC process. Requirement errors always affect the productivity of development, testing team. To make software error free there is need to consider error free customer's SOP. Hence, researcher recommends that before starting development of any software, validate customer's SOP with zero percentage of error.

Most of time even though requirement is error free or correct but design logic mentioned in requirement document is incorrect then it leads to development of incorrect functionality of software product. Around 4.34 percentage of respondents are saying 'Logic Design' is one of the factor responsible for erroneous software product. Design of logic generally comes into Component design document and Functional Requirement Document (FRD). However, if these two documents are incorrect then it always affects to logic of design of functionality. Logic of Design is nothing but of mapping of customers' demands and functionality of software. If this mapping is incorrect then development

team cannot implement exact demand of customer. Hence, researcher recommends that Logic of design is important factor to consider making software error free.

Requirement, development and testing mainly relay on document to complete their work. Requirement team relay on Customer Requirement Document (CRD) to complete Functional Requirement Document (FRD), Business Requirement Document (BRD) and Component Specification document (CSD). Development team also relay on Functional Requirement Document (FRD) to complete their development work and testing team relay on Customer Requirement Document (CRD) to complete Functional Requirement Document (FRD), Business Requirement Document (BRD) to complete their test cases. But suppose all these documents are incorrect then requirement, development and testing will lead to incorrect functionality of software product. Hence, around 4.26% of respondents are saying 'Documentation' is one of the factors that also make software erroneous. Hence, researchers of this research recommend that Documentation of any software project should update, latest and should cover all the functionality of proposed software product.

'Data' factor is also responsible to make software erroneous because software testing is getting done by using dummy data. Most of time software component communicate with other software, machine and for this communication they requires specific data and to make such work possible testing team uses dummy data just make a feel of end to end flow of a software system. Also most of time to validate the functionality of third party device there is need to use correct or appropriate data, and if this data is incorrect then test case might failed for incorrect data. Hence, due to Data factor most of test cases are getting failed. Hence as per around 3.74 percent of

respondents are agrees that Data is one of the responsible factor to make software erroneous. Researcher of this research recommends that Data should always be correct and appropriate to start with software testing.

Environment is nothing but hardware on which software is getting executed. Around 3.74 percentages of respondents are saying environment is also one of the factors responsible for erroneous software. If environment of software is not good then it always fails execution of software. E.g. if there is lack of prerequisite list of softwares or hardware required for proposed software System then in first attempt only software gets failed.

Factors	**SD**	**D**	**N**	**A**	**SA**	**Total**	**Avg**	**Wt.Avg(Likert Scale)**
Logic Design	0	0	17	232	151	400	4.34	4.33
	0	0	4.25	58	38			
Documentation	0	0	64	168	168	400	4.26	4.26
	0	0	16	42	42			
Human	0	0	336	64	0	400	3.16	3.16
	0	0	84	16	0			
Environment	0	0	168	168	64	400	3.74	3.74
	0	0	42	42	16			
Data	0	0	168	168	64	400	3.74	3.74
	0	0	42	42	16			
Interface	0	0	168	232	0	400	3.58	3.58

	0	0	42	58	9			
Requirement Errors	0	0	16	77	307	400	4.73	4.76
			4	19.3	76.8			

Table 4.15: Factors responsible to make software erroneous

'Interface' is nothing but rule or protocol by using that two systems are communicating or sending messages to each other. 3.58 percentages of respondents are saying 'Interface' is a factor which may make software erroneous. Sometime tester can send incorrect message by using such interfaces and in such case software gets failed. Hence test should be aware the format of message or data that interface requires.

'Human' is also factor which makes mistakes in understanding, developing and testing of software product. 3.16 percent of people are saying 'Human' is responsible factor to make software error as in most of manual processes human makes mistakes and it leads to failure of software. Researcher of this research recommends that Resources or Humans should be well trained and skilled to complete the implementation of software product.

4.5.3. Failure of Software Project

21st century known for computerization of all manual works, that human being was doing so far. Computerization made man life easy and this computerization become possible because of integration of hardware and software. Software plays most important role in the automation of most of electronic appliances. Hence, in current market, demand for all types of softwares is increasing day by day. This demand leads to development of thousands of software applications in turn increase in software industries.

Sr. No	Factors	SA(5)	A(4)	N(3)	D(2)	SD(1)	Avg
1	Lack of user involvement	275	91	34	0	0	7.39
2	Long or unrealistic time scale	201	199	0	0	0	7.23
3	Poor or No Customer's SOP	237	156	7	0	0	7.35
4	Inadequate Documentations	158	198	44	0	0	6.88
5	Scope Creep	181	219	0	0	0	7.15
6	No Change Control System	172	191	37	0	0	6.97
7	Poor testing	231	169	0	0	0	7.35
8	Lack of foresight in building efficiency markets	222	141	37	0	0	7.17
9	poor managerial decisions	206	178	16	0	0	7.19
10	Cost overrun.	184	200	16	0	0	7.1
11	Lack of an experienced project manager:	141	236	15	8	0	6.87
12	Lack of methodology in the process	163	230	7	0	0	7.05
13	Well-defined Schedules	238	155	7	0	0	7.35

Table 4.16: Factors responsible for failure of software project

Every year many software industries are spending billion on IT application development. Statistically, 31% of projects will be cancelled before they ever get completed. 53% of projects will cost twice as of their original estimates, overall, the success rate is less than 30% [2]. Why did the project fail? From symptom to root cause -

what are the major factors that cause software projects to fail? What are the key ingredients that can reduce project failure?

Following are the points are considered in this research for the project failure.

1. Lack of user involvement-: In collecting SOP process mainly user involvement is important to gather correct customer's SOP.
2. Long or unrealistic time scale-: Project must be delivered and developed on schedule; if it is too long schedule software can be in failure state.
3. Poor or No Customer's SOP-: If gathered customer's SOP are incorrect or ambiguous then developed software can be in failure state.
4. Inadequate Documentations-: In early phase of software development all customer's SOP and designs must be written in proper format called as documentation.
5. Scope Creep -: It is term used for trimming or missing of customer's SOP during development process of software.
6. No Change Control System-: If changes are suggested by client during development phase and if those are cultivated without change control system, then there is a chance of failure of software.
7. Poor testing-: Proper testing must be there to avoid errors or bugs in software.
8. Lack of foresight in building efficiency markets-: standard must be maintained in other development companies for developing quality product.
9. poor managerial decisions-: Some time wrong managerial decisions are also one of reason for failure of software.
10. Cost overrun-: Unexpected increased cost in budget.

11. Lack of an experienced project manager-: Experience of project manager matters of developing quality product.
12. Well-defined Schedules.-: If software is developed in proper defined Schedule then project can be successful.

It is quite important to understand the factors responsible for software project failure. Table No. 4.16 shows the various factors, which are responsible for software project failure. To meet the objective a questionnaire has been designed by using various factors which define the various points responsible for failure of any software project. It is observed that for each document the average scale is in between 1 to 5 that is in between strongly disagree to strongly agree. In fact all the values are above 3.5 which mean that with respect to all the parameters much approval is observed. In a 5-point Likert scale, having categories like strongly agree, agree, neutral, disagree and strongly disagree clubbed into three categories. The reason for using Likert scale is that the responses by the respondents should not become monotonous while answering the questions. Hence researcher has also applied 5-point Likert scale and calculates weighted average value. There is very less difference between the comparative value of rank order average value and 5-point Likert scale value. [1]

It is seen that the highest average value is 7.39 for the ‘Lack of user involvement’ followed by ‘Poor or No Customer’s SOP’, ‘Poor Testing’ and ‘Well-defined Schedules’ which are 7.35. The average value for factor ‘Long or unrealistic time scale’ is 7.23 followed by ‘poor managerial decisions’ is 7.19. The average value of ‘Lack of foresight in building efficiency markets’ is 7.17, followed by ‘Scope Creep’ is 7.15. The average

value of ‘Cost overrun’ is 7.1 followed by ‘Lack of methodology in the processes is 7.05. The average value of ‘No Change Control System’ is 6.97 and ‘Inadequate Documentations’ is 6.88 followed by ‘Lack of an experienced project manager’ is 6.87.

As per most of respondents, it is clear that ‘Lack of user involvement’ followed by ‘Poor or No Customer’s SOP’ is most important factor responsible for the failure of software project. Off course, user means end user involvement is most important as end user only going to tell his demands or request to business analyst and if end user only unavailable in the collecting SOP meeting then there is no point to discuss anything, anymore. Average 7.39 respondents are agreed to have end user in the collecting SOP meetings or sessions.

Even if end user is available in collecting SOP meeting or session but customer’s SOP quality remains poor then also it leads to failure of software project. Poor customer’s SOP can get collected if business analyst having less domain knowledge. Also if end user does not have understanding what exactly he wants then also quality of requirement becomes poor. Poor or non-qualitative requirement can become base for any software project and it creates failure throughout SDLC process. Hence, in this research 7.39 responds recommended to have good quality of customer’s SOP.

If end user is available and quality of requirement is also good but if testing team executes test cases wrongly then also it creates failures in software project. As per 7.39 respondents, testers should always execute test cases based on the business functionality and customer’s SOP written in Functional Requirement Document (FRD). Hence, in this research recommendation SOP to provide to testing team to follow testing best practices for the test case execution and in turn to reduce software failures.

Project management is the key factor for the success or failure of any software project. For qualitative project management, well-defined schedule is mandatory factor. If project will have well- defined schedule then all the teams like requirement team, development team, designer team, testing team etc. will follow the same time lines to meet the project success. Hence, 7.39 respondents agreed to have well-defined schedule to reduce failure in software project.

As we saw well-defined schedule for software project plays vital role in the success of project, but well-defined schedule mean short and realistic. If project schedule becomes long and unrealistic then it surely leads to failures in software project. About 7.23 respondents agreed to this point and hence researcher strongly recommends that project schedule time scale should be short and realistic.

In software project execution, many situations can come where managerial level people need to take decision and provide answer to client. If project goes in RED situation where customer is not happy and he is demanding software in very short period of time then in that case managerial decision plays very important role to keep customer calm and happy. But if managerial decision becomes poor then customer won't allow us to work and can take break deal with Software Company. Hence, as per average 7.19 people poor managerial decision leads to failures in software project and therefore, there is need to improve managerial decision skills to increase success rate of software project.

If we developed any product then for selling that product marketing plays vital role. But if we don't have foresight about our project efficiency then in the market product won't get sell. The average 7.17 respondents saying 'Lack of foresight in building efficiency markets' is most important factor and software companies need to focus on

this point. Also there is need to have future knowledge about the market of developed software project.

Around average 7.15 respondents are giving importance to 'Scope Creep' factor. Scope creep means project scope should not get trim if we are dealing with success of software project. Scope creep generally happens if project schedule is long and unrealistic. Hence, as discussed above to avoid scope creep there is needed to have well-defined, short and realistic project schedule.

If there is lots of changes in customer's SOP or development team created faulty software component or taken too much time to developed software product, also testing team could not finish testing within specified time then project cost can get overrun. The average 7.1 respondents agreed that 'Cost overrun' could lead to failures in software project. Hence, project management or team led SOP to focus on work status of requirement, development and testing team. Need to resolve all the issues coming throughout SDLC phase so that it cannot overrun cost of the software project.

For error free SDLC process there is standard defined by software engineering for each phase. Requirement team should follow the best standard practices for requirement engineering process, development team should follow best development practices for the coding of software components, and testing team should follow the best testing practices. But if there is lack of methodology present in these best practices then it will lead to project cost overrun and in turn lead to failure of software project. Average 7.05 respondents are agreed with 'Lack of methodology in the process' lead to failure of software project and hence this research recommends best practices and methodology should be followed throughout SDLC process.

To record the changes taken place in customer's SOP by the end user or customer, software management should create change control system and update it as and when required. The average 6.97 respondents agreed that 'No Change Control System' always lead to failure of software project. If project management do not use Change Control System then it won't be possible to record changes made in customer's SOP and it will miss out few important functionality in proposed software project. Hence this research recommends that change control system should be mandatory and gets updated as and when required.

Documentation throughout SDLC process plays vital role to transfer knowledge from one team to another. If requirement team does not provides adequate documents to development team then development team cannot come up with component design and specification documents appropriately and if development team does not provides FRD, component design and specification documents to testing team, then testing cannot come up with appropriate testcases. Hence, each team should provide adequate documents to other team. Because Inadequate Documentation can lead to the failure of software project and in survey around average 6.88 respondents agreeing that 'Inadequate Documentations' lead to software failures and there is need to have adequate documentation throughout SDLC process.

Experienced resources always play vital role in the success of software project. If resources are fresher or new joiners then they don't have product/domain knowledge and hence they cannot understand business functionality easily and quickly. Experienced person can easily communicate with customer on the domain knowledge, business functionality issues etc. Hence, around average 6.87 respondents are agreeing that 'Lack

of an experienced project manager' lead to software failure. Therefore, there is need to have experienced resources in the all the teams of software company.

Information from Table 4.15: Factors responsible to make software erroneous and

1. **Table 4.16: Factors responsible for failure of software project proves third objective *"To identify various factors responsible for the software development".***

4.6. To analyze various tools used in software companies

4.6.1. Useful tools for collecting SOP process.

Collecting SOP is the process to collect demands, SOP, or requirement from client. After collecting such requirement, business analyst SOP to store and manage these collected customer's SOP. In software industry, many tools are available to collect record and manage customer's customer's SOP. For this research, following number of collecting SOP tools have been considered:

1. Visual Paradigms
2. Project Management Software
3. Microsoft-Package
4. Data Dictionary
5. Use Cases and User Stories
6. ReqHarbor
7. MindTool
8. IBM Rational Doors
9. Jira
10. Rally

11. Taleo

12. Quality Center

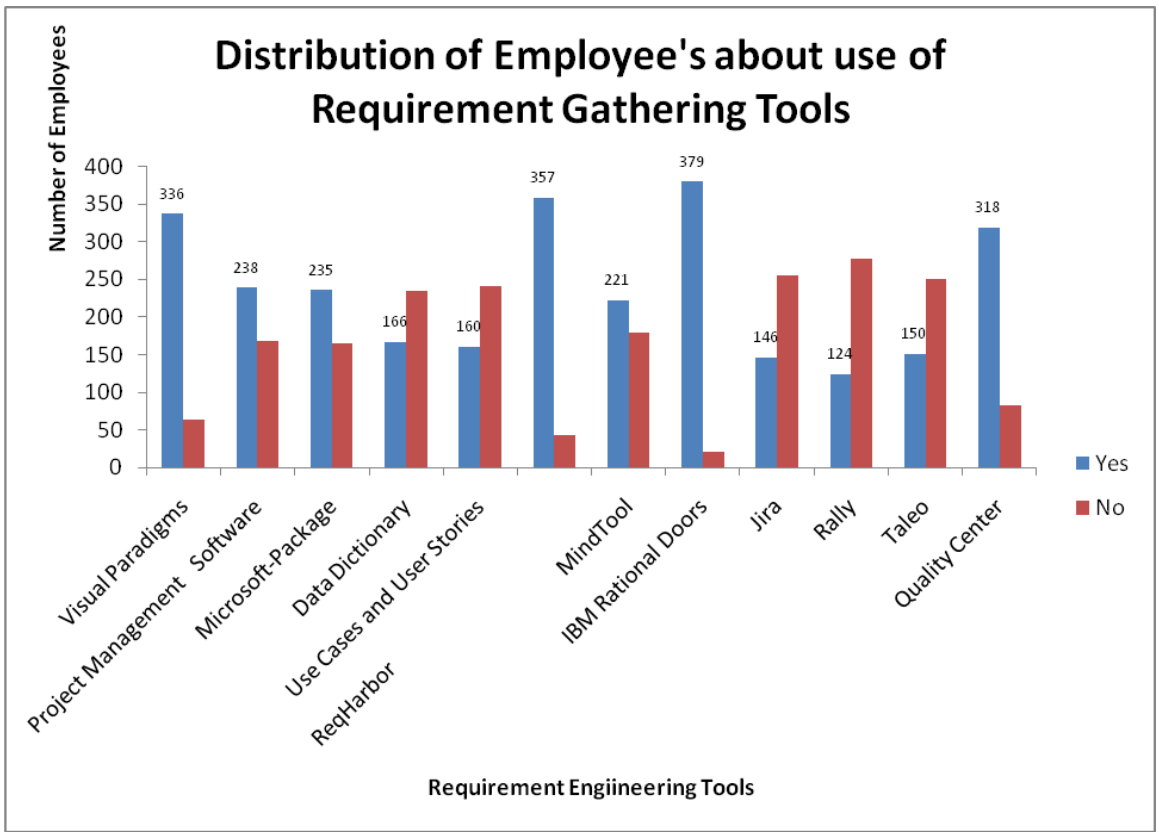
From the survey, it has been seen that 379 respondents are agreeing that 'IBM Rational Doors' is the best collecting SOP tool. Because it supports multiple functionalities for customer's SOP like Collecting SOP, Software Design, Task Management and Collaborative Modeling [10] etc. As IBM Rational Doors all the required functionalities for collected customer's SOP, therefore single software perform the functionality of different software's and hence maximum respondents are agreeing to use 'IBM Rational Doors' tool for collecting SOP, managing and analysis.

The above mentioned information about collecting SOP tools from Table 4.17 and Table 4.6 Collecting SOP techniques proves the second objective.

Collecting SOP Tools	Yes	No
Visual Paradigms	336	64
Project Management Software	238	167
Microsoft-Package	235	165
Data Dictionary	166	234
Use Cases and User Stories	160	240
ReqHarbor.com	357	43
MindTool	221	179
IBM Rational Doors	379	21
Jira	146	254
Rally	124	276

Taleo	150	250
Quality Center	318	82

Table 4.17: Customer's SOP gathering Tools



Graph 4.11: Customer's SOP gathering Tools

As pre 357 respondents, ReqHarbor is also very good collecting SOP tool as it is used to store collection of descriptions of customer's SOP and others who need to refer to them. A first step in analyzing a system of objects with which users interact is to identify each object and its relationship to other objects [7]. Data dictionary basically useful to save requirement related data and anyone can use it for further analysis. Hence, Data dictionary tool plays vital role in requirement collection.

As per 160 respondents 'Use Cases and User Stories' is fundamental factor in requirement management process. Once requirement is collected from client then business analyst maps that requirement with Software Company's product functionality and this mapping is done with the help of usecases. Usecases are nothing but transaction wise functionality of customer demands. User stories are basically segregation of customer demands as per transactions or business functionality changes. Hence usecases and user stories are interrelated to each other and playing most important role in the mapping of customer demands and product functionality.

Around 235 respondents are agreeing that Microsoft-Package is also useful tool for collecting SOP process. As we know that Microsoft-Package is the well-known software package provided by Microsoft Company and it is basically used for documentation, presentation and data analysis in xlsheet. Hence, without Microsoft-Package, most of work of collecting SOP process remains incomplete and therefore almost all companies are using this package to complete requirement analysis. As per 238 respondents, 'Project Management Software' plays vital role in collecting SOP process. Actually senior management who deals with requirement management they use this tools. Hence, 162 respondents who are belongs to managerial category agree to use this tool for requirement management in turn software project management.

4.7. To analyze the current scenario of software testing

Software Testing is a process used to help identify the correctness, completeness and quality of developed computer software. Testing is a process of executing a program with the intent of finding an error. [8]

Testing is a process rather than a single activity. This process starts from test planning then designing test cases, preparing for execution and evaluating status till the test closure.[9]

4.7.1. Type of Software Testing

Generally, in most of software companies, testing of software product is done with the help of following two types:

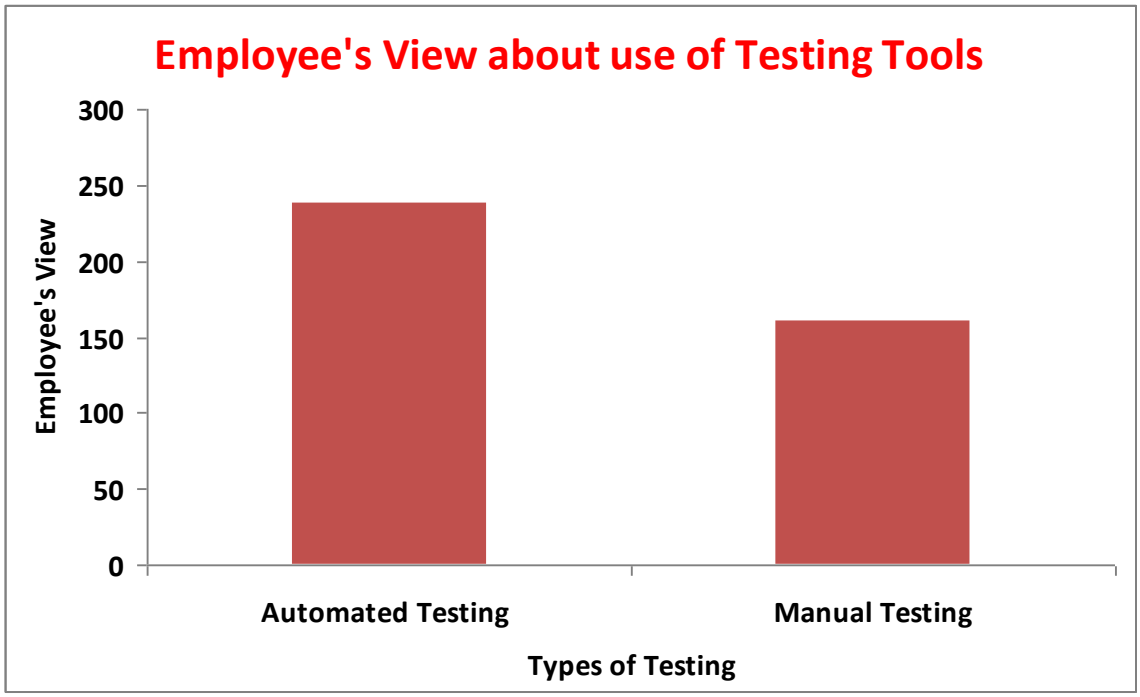
1. Manual Testing
2. Automated Testing

In manual testing, tester SOP to create test case, test data and manually execute test cases with dummy data on particular software component but in case of automated testing, test cases and dummy data has been created by testing tool itself. Hence, in case of automated testing, effort required for test cases execution is getting decrease automatically. Hence, to increase the speed of test cases execution, there is need to use automated testing instead of manual Testing. Hence, in this research survey has been carried out to understand how many tester uses automated testing.

As per survey, total number of respondents who are doing Automated Testing is 239 and the total number of respondents who are doing manual testing is 161.

Which testing type are you preferring?	**Yes**
Automated Testing	239
Manual Testing	161

Table 4.18: Employee's view about use of testing tools



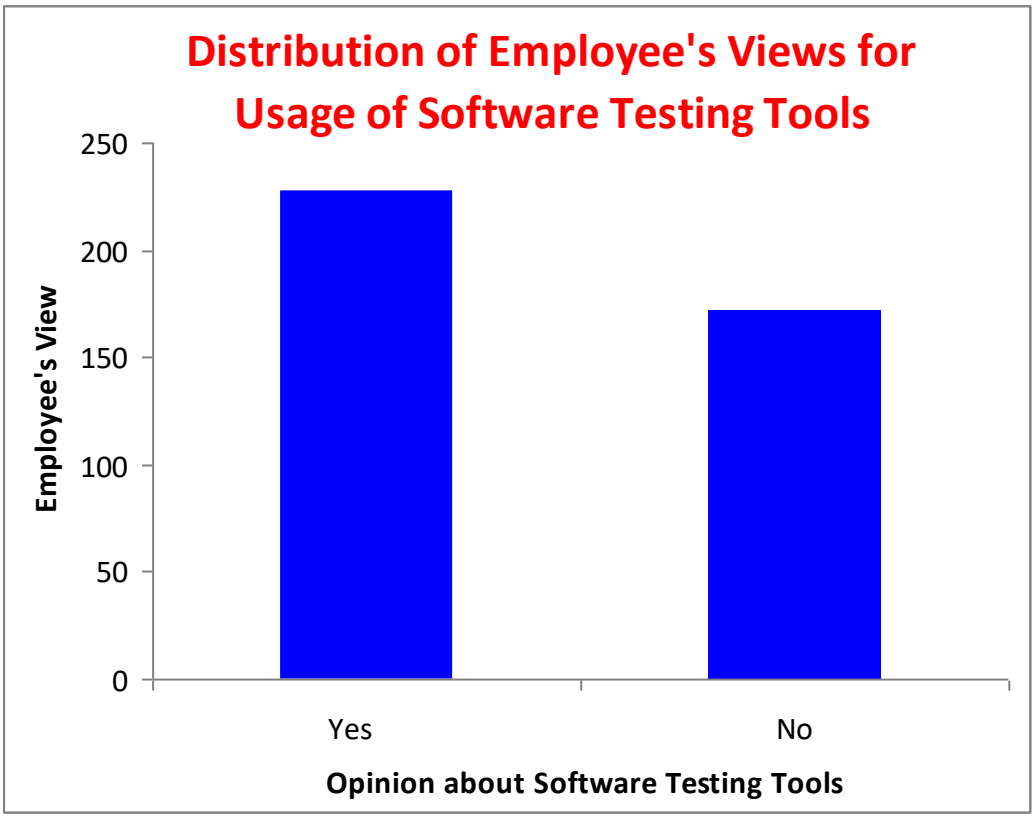
Graph 4.12: Employee's view about use of testing tools

4.7.2. Usage of Software Testing Tools:

As in previous section, researcher has recommended automated testing is good to get better productivity of testing team, hence there is need to know whether testers are using testing tools or not. From the table 4.19, it has been seen that 228 respondents are using testing tools and 172 are still doing testing manually means they are not using testing tools.

Usage	Yes	No
Usage of Testing Tools	228	172

Table 4.19: Employee's view about usage of testing tools



Graph 4.13: Employee's view about usage of automated testing tools

4.7.3. Test Cases

A **test case** is a set of conditions or variables under which a tester will determine whether a system under test satisfies customer's SOP or works correctly.

The process of developing test cases can also help find problems in the customer's SOP or design of an application.

Test Suite ID	The ID of the test suite to which this test case belongs.
Test Case ID	The ID of the test case.
Test Case Summary	The summary / objective of the test case.
Related Requirement	The ID of the requirement this test case relates/traces to.
Prerequisites	Any prerequisites or preconditions that must be fulfilled prior to executing the test.
Test Procedure	Step-by-step procedure to execute the test.
Test Data	The test data, or links to the test data, that are to be used while conducting the test.
Expected Result	The expected result of the test.
Actual Result	The actual result of the test; to be filled after executing the test.
Status	Pass or Fail. Other statuses can be 'Not Executed' if testing is not performed and 'Blocked' if testing is blocked.
Remarks	Any comments on the test case or test execution.
Created By	The name of the author of the test case.
Date of Creation	The date of creation of the test case.
Executed By	The name of the person who executed the test.
Date of Execution	The date of execution of the test.
Test Environment	The environment (Hardware/Software/Network) in which the test was executed.

Table 4.20 Test Case Sample

Execution per day

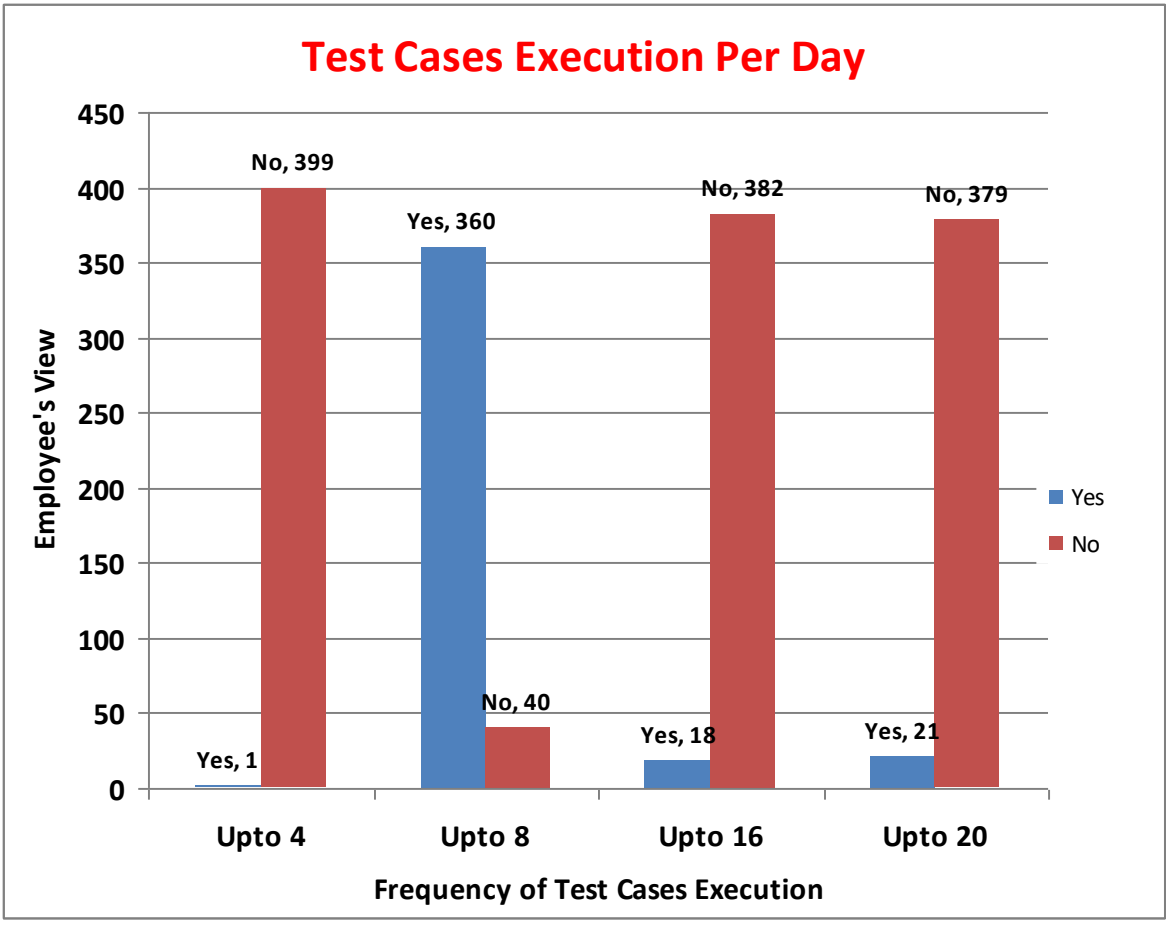
As we saw, Software Testing is done by following two ways: manual testing and automated testing.

As we know that automated testing is quite easier, faster and increase the productivity of testing team, test cases execution per day by using automated testing also increases the count of testing. But using manual testing as it is manual process and requires lots of human intervention, hence, test cases execution using manual testing giving quite low count.

Hence in this research survey has been carried out mainly on the test cases execution per day using automated testing mode. As per 1 resource 4 test cases are executed per day which is quite low number of agreed view of employee. But maximum around 360 employees are agreeing that upto 8 testcases can be executed per day. And yes it is correct count because 8 test cases executed per day it is standard count of software testing. So as per survey it is recommended that execution of 8 testcases per day using automated testing mode is best practice for software testing process. But in case of testing scenarios are easy or there is urgency from client then in that case tester can test upto 16 or 20 testcases in one day.

No of Test Cases	Yes	No
Upto 4	1	399
Upto 8	360	40
Upto 16	18	382
Upto 20	21	379

Table 4.21: Test cases execution per day



Graph 4.14: Test cases execution per day

4.7.4. Defects raised by testing team per day

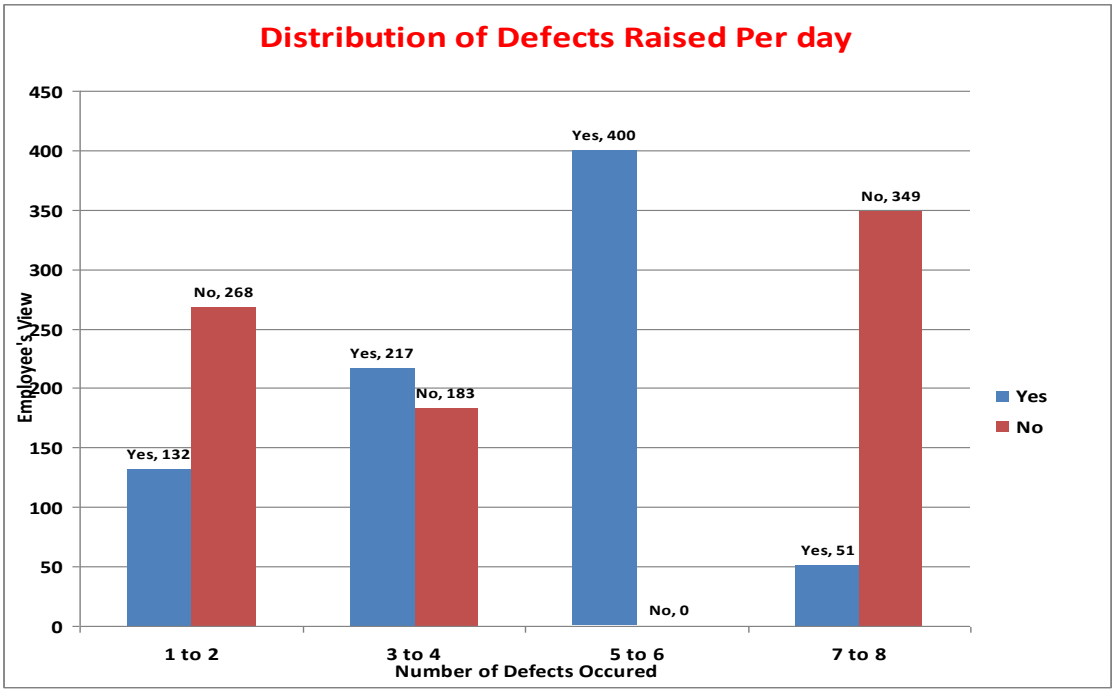
- A defect is an error or a bug, in the application which is created. A programmer while designing and building the software can make mistakes or error. These mistakes or errors mean that there are flaws in the software. These are called defects.

As we saw, using automated software testing mode, tester can execute 8 testcases per day. However, complete execution of these 8 testcases is basically depends upon on their test results. If test result is successful then it means there is no need to re execute test case

but if test result is fail then re-execution becomes mandatory. If test cases are failed it means testers are raising defects against these failed test cases. Moreover, per day how many defects can be raised by testing team becomes important for management in terms of project completion.

Frequency of Defects raised per day	Yes	No
1 to 2	132	268
3 to 4	217	183
5 to 6	400	0
7 to 8	51	349

Table 4.22: Defects raised per day



Graph 4.15: Defects raised per day

In this research, survey has been done on testers who are doing automated software testing as per 132 testers 1 to 2 defects can occur per day. Around 217 respondents are saying 3 to 4 defects are occurring per day and maximum testers like almost all tester means 400 testers are saying 5 to 6 defects are occurring per day. However, the chances of occurring 7 to 8 defects per day are quite low as per 51 respondents. Hence, researcher has recommended that tester should expect 5 to 6 defects per day in their software testing process.

4.7.5. Significance of document in software testing process

As we saw in section 4.6, in collecting SOP session or meeting, business analyst SOP to take customer's SOP verbally from client and then he/she SOP to record or write requirement in specific document. Recording customer's SOP somewhere in document is a need of an hour because for further changes or future use we need base of requirement. Hence, to record such customer's SOP, software engineering designed and defined different documents likewise software testers also need to consider following list of documents in the software testing process to cover the testing volatile functionality of software product. Based on requirement documents Testers always need to create test cases document and need to perform the testing as per test cases. Following are the documents that testing team need to consider to enhance the performance of software testing process.

1. Customer Requirement Document (CRD)
2. Business Requirement Document (BRD)
3. Functional Requirement Document (FRD)
4. Component Specification Document (CSD)

5. Component Design Document (CDD)
6. Test Case Document (TCD)

It is quite important to understand, among six documents which document is playing most vital role in the success of software testing. Table No. 4.23 shows the various documents used in SDLC process. To meet the objective a questionnaire has been designed by using various documents, which define the various documents responsible or useful for software testers. It is observed that for each document the average scale is in between 1 to 5 that is in between strongly disagree to strongly agree. In fact all the values are above 3.5 which mean that with respect to all the parameters much approval is observed. In a 5-point Likert scale, having categories like strongly agree, agree, neutral, disagree and strongly disagree clubbed into three categories. The reason for using Likert scale is that the responses by the respondents should not become monotonous while answering the questions. Hence researcher has also applied 5-point Likert scale and calculates weighted average value. There is very less difference between the comparative value of rank order average value and 5-point Likert scale value. [1]

It is seen that the highest average value is 6 for the 'Test Case Document (TCD)' followed by 'Functional Requirement Document (FRD)' and 'Component specification Document (CSD)' that is 5. The average value of 'Component Design document (CDD)' is 4.8 followed by Customer Requirement Document (CRD) that is 4.22. And last is 'Business Requirement Document (BRD)' that is 4.13.

It is clear from the average values that Test Case Document (TCD) is most important document for testing team as per most of respondents. Test Case Document

(TCD) is mainly important document for testing team only. But most of time designer team creates this document for testing team. And once testing team finish their testing then designer team verifies test results with test case document. For testing team test cases document always plays very important role because it is the base on which whole testing team actually works. Hence as per average 6 % respondents are saying that test case document is most valuable document for testing team and playing significance role in the failure of software product. Because test case document is created by referring Functional Requirement Document (FRD) and hence test case document should always be in sync with FRD. And if any test case is not getting matched with required functionality written in FRD then it might lead to incorrect direction for software testing process and lead to wastage of time. Hence in this research researcher recommends that test case document should get created correctly and gets followed as per FRD. Also it would be great if this document is created by designer team because designer team has complete knowledge about client requirement and they can include complete functionality scenarios in test case document.

As FRD is base for test case document and it is created to include the functionality of proposed software and it is useful for Testing, development and designer team as well. Hence, most preference has been given to this document. FRD always contains the detail information about integrated functionality of proposed software and customer's demands or requirement in the technical language. Hence, FRD should always be in sync with updated requirement and updated functionality of domain. Component design document (CDD) actually contains the information about technical, business functionality developed component of proposed software, and hence this document is mainly useful for

development and testing team. Testing team uses CDD document just to understand business functionality in technical terms. Component Specification Document (CSD) is needed for designer and development team and hence employees from both these teams have given preference to this document. Test Case Document (TCD) is mainly important document for testing team and hence this document should be in sync with FRD that is the business functionality of proposed document. Customer Requirement Document (CRD) is the base for all documents. This document actually created by customer itself and accordingly business analyst and designer creates BRD, FRD, and other documents. CRD always contains customer's specific demands or customer's SOP and it is available in the customer's understanding language. BRD is created by considering CRD and it contains only business functionality of proposed software. It does not have technical functionality and hence this document is only helpful for business analyst and system architecture team.

Documents	**SD**	**D**	**N**	**A**	**SA**	**Total**	**Avg**	**Wt.Avg Likert Scale**
Customer Requirement Document (CRD)	0	15	42	183	160	400	4.22	4.3
	0	(3.75)	(10.5)	(45.75)	(40)	100		
Business Requirement Document (BRD)	0	0	112	126	162	400	4.13	4.13
	0	0	(28)	(31.5)	(40.5)	100		
Functional Requirement Document	0	0	0	0	400	400	5	5
	0	0	0	0	(100)	100		

(FRD)								
Component Spécification	0	0	0	0	400	400	5	5
Document (CSD)	0	0	0	0	(100)	100		
Component Design	0	0	15	51	334	400	4.8	4.8
Document (CDD)	0	0	(3.75)	(12.75)	(83.5)	100		
Test cases Document	0	0	0	147	253	400	6	4.63
(TCD)	0	0	0	(36.75)	(63.25)	100		
Total	0	0	169	507	1709			
Avg. Percentage	0	0.63	7.04	21.13	71.21			

Table 4.23: Significant Documents used in Software testing process

4.8 To understand the various hurdles coming in the software testing and testers problem.

4.8.1. Cost considered during software testing process

As we have saw, many factors are responsible for software project failure. Failure of software project mainly being calculated based on its cost. If cost of software getting decrease it means surely failure is there in software. Table No. 4.24 shows the various factors, which are responsible for software cost and in turn project failure. To meet the objective a questionnaire has been designed by using various factors, which define the

various points responsible for cost of any software project. It is observed that for each factor the average scale is in between 1 to 5 that is in between strongly disagree to strongly agree. In fact all the values are above 3.5 which mean that with respect to all the parameters much approval is observed. In a 5-point Likert scale, having categories like strongly agree, agree, neutral, disagree and strongly disagree clubbed into three categories. The reason for using Likert scale is that the responses by the respondents should not become monotonous while answering the questions. Hence, researcher has also applied 5-point Likert scale and calculates weighted average value. There is very less difference between the comparative value of rank order average value and 5-point Likert scale value. [1]

It is seen that the highest average value is 6.784 for the 'Software' followed by 'Resources' which is 6.02. The average value for factor 'Hardware' is 5.916 followed by 'Network' is 5.76. The average value of 'Infrastructure' is 4.884.

In this research, survey has been done and as per average values of respondents views 'Software' is the biggest cost considered during the failure in software testing process. Software is nothing but product that is going to fulfill customer's demands or customer's SOP and if due to failures in software testing off course software product deadly gets affected and hence software SOP to be considered at high priority cost when there is failure in software testing process. Failures in software testing process always affect the quality of developed software and hence, software would become highest cost of software failures.

After software, resources become next high priority cost factor to consider in failure of software testing process. Because if software testing gets failed then it totally affects

on the productivity of testing team and creates extra efforts for them and hence respondents provided second highest priority for resources in term of software cost.

Factors (utilization)	**SD(1)**	**D(2)**	**N(3)**	**A(4)**	**SA(5)**	**Total**	**Avg**	**Wt.Avg Lokert Scale**
Resources	0	1	19	64	316	400	6.02	4.74
	0	0.25	4.75	16	79	100		
Software	0	1	41	219	139	400	6.784	4.25
	0	0.25	10.25	54.75	34.75	100		
Hardware	0	1	198	122	79	400	5.916	3.7
	0	0.25	49.5	30.5	19.75	100		
Network	0	1	178	101	100	380	5.76	3.61
	0	0.25	44.5	0.25	25	70		
Infrastructure(electricity, rent etc)	0	41	197	62	60	360	4.884	3.26
		10.3	49.3	15.5	15			
Total	0	45	633	568	694			
Avg. Percentage	0	31.66	23.4	34.7	74			

Table 4.24: Cost Factors involved in testing process in terms of project failure

As shown in Table 4.24, the average value of Hardware factor followed by resources and as per respondent around average 5.916 respondents are agreed that Hardware is also one of the important factor to be consider as a cost in software failures. Because if any software product gets failed then hardware also getting affected as number of resources are uses these hardware to run their software. Hence, if software gets failed then obviously it effects on the efficiency and productivity of hardware. Hence, while considering cost of factors of software failure, management should consider hardware also.

In software industry, network plays vital role in data transfer. And such data can be transferred from client to software provider or from software provider to client. To use network related data transactions, software industrialist always SOP to pay money to internet service provider and if any developed software gets failed then off course it gets affects to Network as well. Because if size of software becomes huge then it may be create congestion in network traffic and lead to failed network flow. Hence while considering software failure cost, network factor should also get considered. As per around 5.76 respondents are agreeing to these points.

Last but not least, infrastructure also plays very important role in overall development of software product and in turn software industry and hence it also gets affected when particular software gets failed. Infrastructure covers overall need of software product and hence it is quite important to consider in software failure. Around 4.884 respondents are agreeing to this point.

4.8.2. Impact of collecting SOP on software testing process

If collected SOP are not clear in the collected customer's SOP from client or customer, is nothing but erroneous requirement. As we saw in the section of failure of software process, poor requirement always affect complete SDLC cycle and hence in software testing point of view, there is need to understand what kind of work actually getting affected mainly in software testing team. Software testing team mainly deals with following list of tasks:

1. Addition of test case
2. Deletion of test case
3. Modification of test case
4. Re-execution of test case
5. Verification of newly added functionality due to requirement change
6. Test results creation for newly added requirement

Above listed tasks of testing team are well known in software industry. Hence, it is quite important to understand, among six tasks which task is getting impacted most due to poor customer's SOP. Table No. 4.25 shows the various tasks performed by software testing team. To meet the objective a questionnaire has been designed by using various tasks of software testing team. It is observed that for each task has the average scale in between 1 to 5 that is in between strongly disagree to strongly agree. In fact all the values are above 3.5 which mean that with respect to all the parameters much approval is observed. In a 5-point Likert scale, having categories like strongly agree, agree, neutral,

disagree and strongly disagree clubbed into three categories. The reason for using Likert scale is that the responses by the respondents should not become monotonous while answering the questions. Hence researcher has also applied 5-point Likert scale and calculates weighted average value. There is very less difference between the comparative value of rank order average value and 5-point Likert scale value. [1]

It is seen that the highest average value is 4.62 for the 'Addition of test cases' followed by 'Modification of test cases' is 4.47. The average value of 're-execution of modified test cases' is 4.46 followed by 'Test result creation for newly added test cases'. The average value of 'Deletion of test case' is 4.16 followed by 'Verification of Newly added functionality due to Requirement Change' that is 4.

Factors	**SD**	**D**	**N**	**A**	**SA**	**Total**	**Avg**	**Wt.Avg (Likert Scale)**
Addition of Test Case	0	0	0	153	247	400	4.62	4.6
	0	0	0	38.25	61.75			
Deletion of Test Case	0	0	61	215	124	400	4.16	4.15
	0	0	15	53.75	31			
Modification of Test Case	0	0	0	214	186	400	4.47	4.46
	0	0	0	53.5	46.5			
Re-Execution of Modified Test Cases	0	0	0	215	185	400	4.46	4.5
	0	0	0	53.75	46.25			

Verification of Newly added functionality due to Requirement Change	0	0	62	276	62	400	4	4
	0	0	16	69	15.5			
Test Results creation for newly added requirement	0	0	0	246	154	400	4.39	4.39
	0	0	0	61.5	38.5			

Table 4.25: Work of Software testing process

If there is change in requirement or requirement is incorrect then it might affect to the complete list of above tasks. Test case creation, review and update in test cases are basic activity of software testing team. While writing test cases, test team follows Functional Requirement Document (FRD) and as we know that FRD is nothing but one of the requirement document created by requirement team. But if FRD is incorrect or incomplete then test cases created by testing team can also be incorrect because incorrect FRD obviously lead to incorrect or incomplete test cases. Incorrect or incomplete customer's SOP means poor requirement and if requirement is poor then definitely it affects on the work of software testing team.

As per 4.62% of respondents saying that "Addition of test cases" is most frequent task tester SOP

to do when requirement is poor. Because when requirement is incomplete, then testing team SOP to add test cases in the existing list of test cases to cover the newly added customer's SOP.

Also around 4.47% respondents are agreeing that "Modification of test cases" task is frequent task done by testers when requirement is poor because if requirement is incorrect then after correcting of requirement, testing team need to update existing test cases as per updated customer's SOP.

Along with test case addition or modification, around 4.46% respondents are focusing on 'Re-execution of modified test cases' because once test cases added or updated as per updated requirement then testing team starts re-execution of newly added or updated test cases to verify the updated functionality mentioned in revised requirement document.

Along with test cases creation, testing team always need to do test results as well. Around 4.39% of respondents are saying after re-execution of newly added test cases, testing team need to create test results as well.

Once addition, modification and re-execution of test cases done by testing team then there is needed to verify all the executed test cases along with their test results. Hence, around 4% or respondents are saying 'Verification of Newly added functionality due to Requirement Change' is important work that designer and testing team lead need to do due to poor customer's SOP.

Researcher of this research recommends that designer team and management always need to focus on to reduce errors in requirement documents so that further phase of SDLC will not get impacted and it will not lead to failure of software project.

4.8.3. If collected SOP are not freezed, it has impact on software development process.

Above table 4.25 shows that change in need for software development has impact on modifying test case. Modifying test case is again counted in terms of total efforts required to this task, again efforts are nothing but it has impact on business.

For this, data is collected through from 400 employees regarding opinion about impact of change in SOP on all task of software development process. Efforts required for task are considered as Development effort, Rework effort, Quality Assurance effort, Testing Effort.

"Best Practices for Change Impact Analysis" article on Impact Analysis for Requirement change by Karl Wiegers at Jama Software's on February 19, 2014.[13]

In this article author has explained the concept of requirement change and if change is given by client, how impact analysis technique can be used. In this article he has explained the format of recording change in the document termed as "Proposed Change" and technique termed as Impact Analysis discussed and then total efforts calculated based on proposed change document.

Effort (Labor Hours)	Task
_____	Update the SRS or requirements database.
_____	Develop and evaluate a prototype.
_____	Create new design components.
_____	Modify existing design components.
_____	Develop new user interface components.
_____	Modify existing user interface components.
_____	Develop new user documentation and help screens.
_____	Modify existing user documentation and help screens.
_____	Develop new source code.
_____	Modify existing source code.
_____	Purchase and integrate third-party software.
_____	Modify build files.
_____	Develop new unit and integration tests.
_____	Perform unit and integration testing after implementation.
_____	Write new system and acceptance test cases.
_____	Modify existing system and acceptance test cases.
_____	Modify automated test drivers.
_____	Perform regression testing.
_____	Develop new reports.
_____	Modify existing reports.
_____	Develop new database elements.
_____	Modify existing database elements.
_____	Develop new data files.
_____	Modify existing data files.
_____	Modify various project plans.
_____	Update other documentation.
_____	Update the requirements traceability matrix.
_____	Review modified work products.
_____	Perform rework following reviews and testing.
_____	Other additional tasks.
_____	**Total Estimate Effort**

Fig. 4.1 Estimated Efforts for change in collected SOP

Software development effort estimation is the process of predicting the most realistic amount of effort (expressed in terms of person-hours or money or resource cost) required to develop or maintain software based on incomplete, uncertain and erroneous SOP from customer. [12] .When SOP from customers are volatile or keep on changing then Efforts of employees affected most because whichever task initially done by employees (earlier efforts) must be changed, so again employees are doing same work as per suggestions means development efforts i.e coding task , Rework effort i.e redesigning of product, Quality Assurance effort i.e product must be measured for its better quality, Testing efforts i.e whichever code has been changed by coder or programmer must be tested

again by writing test cases again (Refer table 4.25) , means all these efforts must be carried out for changes nothing but it has impact on cost which can be counted as “Impact on Business”.

Sr.No	Efforts	Yes	No
1	Development Efforts	400	0
2	Rework Efforts	339	61
3	Quality Assurance Efforts	376	24
4	Testing Efforts	400	0
	Total	1515	85
	Avg	378.75	21.25

Table 4.26: Efforts carried out in case collected SOP are not freezed

In the above table 4.26 researcher found that most of the employees are strongly agreed on Development and testing efforts must be carried out which requires extra cost during development of software.

4.8.4. Overhead occurs in software testing due to incomplete or ambiguous SOP collected from customer.

Collecting SOP is the process where incorrect, incomplete, ambiguity, vague, volatile requirement issues can occur. Due to issues present in customer’s SOP, it always affects to software testing team productivity because issues create discontinuity in software testing process. Due to issues present in customer’s SOP, many overheads can occur in

software testing process. For this research, following list of overheads has been considered:

1. GAP in Testing
2. Increase in System Failures
3. System Testing Delay
4. Inaccurate Testing Estimation
5. Test Team Credibility
6. Delay Benefit Realization

An overhead is the extra burden on testing team and it always decreases the productivity of testing team. Above listed overheads are mostly found as factors which affects software testing productivity. Hence, it is quite important to understand which overhead is faced by testing team most. Table No. 4.27 shows the various overheads that software testing team face. To meet the objective a questionnaire has been designed by using various overheads that creates most burdens on software testing team. It is observed that for each overhead has the average scale in between 1 to 5 that is in between strongly disagree to strongly agree. In fact, all the values are above 3.5, which mean that with respect to all the parameters much approval is observed. In a 5-point Likert scale, having categories like strongly agree, agree, neutral, disagree and strongly disagree clubbed into three categories. The reason for using Likert scale is that the responses by the respondents should not become monotonous while answering the questions. Hence researcher has also applied 5-point Likert scale and calculates weighted average value.

There is very less difference between the comparative value of rank order average value and 5-point Likert scale value. [1]

It is seen that the highest average value is 4.93 for the 'Gap in testing' followed by 'inaccurate testing estimation' is 4.27. The average value of 'System Testing Delay' is 4 followed by 'Test Team Credibility' is 3.8. The average value of 'Increase in system failures' is 3.6 followed by 'Delay benefit realization' is 3.59.

If there is issue in requirement then it affects to the software testing as discontinuity in their work. Because if testing finds issue in customer's SOP then they are raising defects in tracking system against requirement team and pause their work until testing team got solution from requirement team and due to this issue 'Gap in Testing' occurs. Same case can happen with development team. Around 4.93% of respondents are agreeing that poor customer's SOP always create in gap in testing work of testers.

Gap in testing or discontinuity in tester work lead to wastage of time of testing resources and it lead to Inaccurate testing estimation. Because if test manager provides initial estimation by considering accurate SDLC process but once any issue comes in customer's SOP or development then it lead to time investment in issue recording, discussion with requirement and development team. Ultimately, it affects of original estimation provided by test manager and hence leads to inaccurate testing estimation. Hence, around 4.27 % of respondents are saying inaccurate testing estimation is one of the overhead that software testing team facing. Hence, researcher of this research recommends that Test manager should keep buffer in testing effort estimation. Buffer is nothing but extra time that test manager need to consider while providing test effort estimation.

Due to issues in customer's SOP, most of time testing team SOP to revise their testing plan. If many issues comes in customer's SOP or development then for each and every issue, tester need to create entry for the same in recording system, also they need to communicate about issues with requirement and development team. After communication of issue, testing team need to wait for the solution of raised issue. In this process, testing team need to wait a lot and hence, such process leads to delay in System testing. In this research, around 4 of respondents are saying 'System Testing Delay' is one of overhead that testing team is generally facing and hence research recommends that higher manager SOP to consider all the pitfalls that testing team actually facing.

As we saw, due to issues present in developed components, testing team need to follow complete process of issue recording to discussion with corresponding team members. To complete such process testing team need to consume lots of time and which in turn leads to decrease in credibility of testing team. Hence, around 3.8% respondents are saying Test team credibility is one of overhead that testing team always facing. Hence, researcher of this research recommends that management should consider credibility of testing team while providing test estimation plan.

As we saw, if collected SOP are wrong then it leads to failure of all the phases of SDLC process. Failure in SDLC process means failures in overall software system. Every time if requirement is incorrect or incomplete then it leads to increase in software system failures. Hence around 3.6% agreeing that 'Increase in system failures' is also one of the overhead that testing team is facing. Hence, researcher recommends that trained and skilled business analysts should be hired.

If there are issues in customer's SOP or developed software components then it will always lead to delay in software testing and in turn delay in project delivery. Once software project got delayed then management miss the realization of product benefit. And as per 3.59% of respondents are saying 'Delay benefit realization' is also one major overhead that testing can face. Hence, researcher of this research recommends that filtration of issues in customer's SOP or development should be happened before delivery of software product to testing team.

Overheads	**SD**	**D**	**N**	**A**	**SA**	**Total**	**Avg**	**Wt.Avg**
GAP in Testing	0	0	0	27	373	400	4.93	4.93
	0	0	0	6.75	93.25	100		
Increase in System Failures	0	0	186	187	27	400	3.6	3.6
	0	0	46.5	46.75	6.75	100		
System Testing Delay	27	0	80	132	161	400	4	4.3
	6.75	0	20	33	40.25	100		
Inaccurate Testing Estimation	0	0	80	134	186	400	4.27	4.26
	0	0	20	33.5	46.5	100		
Test Team Credibility	27	0	133	106	134	400	3.8	4
	6.75	0	33.25	26.5	33.5	100		
Delay_benefit_relaisation	54	27	54	160	105	400	3.59	4.26
	13.5	7	13.5	40	26.5			

Table 4.27: Overhead occurrences in software testing due to poor collecting SOP

4.8.5. Common Requirement Issues that may affect Software Testing

Software is developed according to Clients Customer's SOP. Here some requirement issues are discussed with software developer and tester, which may affect software-testing process. Design or requirement issues can occur in any phase of SDLC process. The possibility of rework due to requirement issues or defect can be minimum if these issues are getting solved in requirement or development phase itself but if requirement issues comes in testing phase then it affects testing, requirement and development phase as well. In this research, following parameters has been considered as software testing issues.

1. Absence and Incompleteness
2. Incorrectness
3. Ambiguity and Vagueness
4. Volatility
5. Traceability

In table 4.27 all the above issue factors are mentioned and it is observed that for each parameter the average scale is in between 1 to 5 that is in between strongly disagree to strongly agree. In fact, all the values are above 3.5 which mean that with respect to all the parameters much approval is observed. In a 5-point Likert scale, having categories like strongly agree, agree, neutral, disagree and strongly disagree clubbed into three categories. The reason for using Likert scale is that the responses by the respondents should not become monotonous while answering the questions. Hence researcher has also applied 5-point Likert scale and calculates weighted average value. There is very less

difference between the comparative value of rank order average value and 5-point Likert scale value.

Factors	**SD**	**D**	**N**	**A**	**SA**	**Total**	**Wt. Avg**	**Wt. Avg (Likert Scale)**
Absence and Incompleteness	0	0	0	110	290	400	4.73	4.72
	0	0	0	27.5	72.5			
Incorrectness	0	0	60	70	260	400	4.4	4.4
	0	0	15	17.5	65			
Ambiguity and Vagueness	0	0	80	90	230	400	4.38	4.37
	0	0	20	22.5	57.5			
Volatility	0	0	60	80	260	400	4.5	4.5
	0	0	15	20	65			
Traceability	0	40	60	70	230	400	4.23	4.2
	0	10	15	17.5	57.5			
Total	0	40	260	420	1270			

Table 4.28: Common Requirement Issues that may affect Software Testing

It is seen that the highest average value is 4.73 for the ‘Absence and Incompleteness’ followed by ‘Volatility’ is 4.5. Average value of ‘Incorrectness’ is 4.4

followed by ‘Ambiguity and Vagueness’ that is 4.38. In addition, the average value of ‘Traceability’ is 4.23.

Requirement issue always affects all the phases of SDLC process. As per research survey ‘Absence and Incompleteness’ is the main factor considered by 4.73 percentages of respondents. If requirement is incomplete or important part is absent in requirement document then it might lead to failure in all the phases of SDLC. Incomplete requirement always lead to incomplete functionality and delay in project delivery to client. The reason behind incomplete requirement is delay from customer or business analyst may have lack of product knowledge. Hence as per most of respondent Incomplete or absent requirement is the root cause of software failure. Hence, researcher of this research recommends that requirement should be complete or should not be missing any important part.

Volatility is nothing but changes in requirement. Volatility mainly happens due to changes in customer’s demands or lack of product knowledge of requirement analyst. Requirement plays base role for all phases like development, testing etc. Because development team works on the base of requirement document but if, requirement document frequently is changed then development team need to rework according to changes in requirement document. Changes in requirement also affect the testing phase as well. If requirement gets changed then testing team also need to retest all the test cases and need to verify updated functionality. Most of time testing team SOP to add test cases as well for updated customer’s SOP. Hence, updated requirement always lead to rework for development and testing team. In turn, volatility leads to increase the workload for development and testing team, and it leads to decrease productivity of both teams. If

productivity decreases then it increases failure in software productivity. Hence, most of respondents are agreeing that volatility is the second most important factor or issue of requirement. Researcher recommends that volatility need to consider for failure of software project and need to ask customer to give requirement in one go only.

As per many respondents, incorrectness is also important factor in requirement issues. Incorrectness generally happens due to inadequate knowledge of business analyst. Business analyst does not understand complete functionality of software product and hence they cannot map customer's demands and software product functionality. Most of time, Incorrectness occurs due to incomplete of requirement or missing or absent of important part in requirement document. Incorrect requirement always lead to development and verification of incorrect functionality and in turn it affects to productivity of development and testing team. In addition, as we know decrease in resources productivity always increases failures in software productivity. Hence, researcher recommends here business analyst always need to verify customer's SOP correctness before delivering it to development and testing team.

Ambiguity and vagueness issue occur due to lack of product knowledge. Most of business analyst having less experience and hence they don't aware about the complete functionality of software product. Hence, they cannot map customer's demands with functionality of their software product. Due to lack of product knowledge, requirement becomes vague and ambiguity. But if requirement becomes vague and ambiguity then development and testing team SOP to consume more time on their work. It requires more time for requirement understanding and implementation. Testing team also need to consume more time on requirement understanding and test cases creation. In this

research, most of respondents are agreeing that ambiguity and vagueness are also important requirement issue to be considered in the failure of software product and software testing. Hence, researcher recommends that if requirement is ambiguity or vague then tester should raise defect/voice against requirement team to revise requirement document with mapping of correct customers demands and functionality of software product.

If requirement is volatile then there is need to keep track of each and every change for the betterment of SDLC process. So that testing and development team can verify component functionality as per latest changes made in requirement document. To keep such changes there is need to have common place so that requirement, development and testing team can have easy access of this place. In software engineering such place is known as traceability. Traceability always gets updated by requirement team if requirement gets change. But if any change is getting miss in the traceability sheet then it might lead to incompleteness of requirement and as we saw above, incompleteness of requirement leads to failure of software product. Hence researcher recommends that traceability sheet should always be in sync with updated requirement of customer. Also this sheet should get review by team lead or development and testing manager time to time so that they can catch missed requirement change.

Information from Table 4.21(Test cases execution per day), Table 4.22(Defects raised per day),Table 4.23: Significant Documents used in Software testing process, Table 4.25: Work of Software testing process, Table 4.26: Overhead occurrences in software testing due to poor collecting SOP, Table 4.27: Common Requirement

Issues that may affect Software Testing proves fourth objective *"To study impact of Collecting needs from customer on business of IT companies".*

4.8.6. Common SOP Issues that may affect Software business.

For this, researcher has collected data from 10 companies, where every company's 5 clients details have been collected for measuring development time or duration span during year 2015-2016.

According to SDLC phase, first phase where client's SOP is finalized. 30 % to 35 % time of total development time is only required to collect SOP and freezed it. [14]

From each company 5 clients data is collected and analyzed their SOP collection duration. From table 4.29 researcher has observed that for each company out of 5 clients atleast 3 clients are taking more time for SOP as their SOP is not freezed and eventually it has impact on software business.

Sr.No	Company Name	Client Name	Total Project Duration (in Months)	Actual Duration required for SOP (in months)	% in months duration only for SOP	Impact on COST when SOP is not freezed				Total No of Clients took extra time to complete SOP
						upto 10%	upto 20%	upto 30%	upto 40%	
1	KPIT Cummins	Client A	10	4	40		Yes			3
		Client B	15	5	33.33	Yes				
		Client C	24	10	41.66667		Yes			
		Client D	12	4	33.33333	Yes				
		Client E	8	4	50			Yes		
						upto	upto	upto	upto	5

						10%	**20%**	**30%**	**40%**	
2	**SAP**	Client A	18	8	44.44444			**Yes**		
		Client B	**10**	**5**	**50.00**				**Yes**	
		Client C	**36**	**15**	**41.66667**			**Yes**		
		Client D	**20**	**8**	**40**			**Yes**		
		Client E	**12**	**5**	**41.66667**			**Yes**		
						upto 10%	**upto 20%**	**upto 30%**	**upto 40%**	
3	**Harman**	Client A	**15**	**4**	**26.66667**					
		Client B	**20**	**8**	**40.00**		**Yes**			**3**
		Client C	**16**	**8**	**50**			**Yes**		
		Client D	**10**	**4**	**40**		**Yes**			
		Client E	**12**	**4**	**33.33333**	**Yes**				
						upto 10%	**upto 20%**	**upto 30%**	**upto 40%**	
4	**Intelzign**	Client A	10	4	40		**Yes**			
		Client B	**18**	**6**	**33.33**	**Yes**				**3**
		Client C	**24**	**11**	**45.83333**		**Yes**			
		Client D	**14**	**5**	**35.71429**	**Yes**				
		Client E	**12**	**5**	**41.66667**		**Yes**			
						upto 10%	**upto 20%**	**upto 30%**	**upto 40%**	
5	**L & T Infotech**	Client A	12	4	33.33333	Yes				
		Client B	**18**	**8**	**44.44**		**Yes**			**3**
		Client C	**20**	**10**	**50**			**Yes**		
		Client D	**15**	**6**	**40**		**Yes**			
		Client E	**10**	**3**	**30**	**Yes**				
						upto 10%	**upto 20%**	**upto 30%**	**upto 40%**	
6	**ATOS**	Client A	24	10	41.66667		Yes			
		Client B	**15**	**5**	**33.33**	**Yes**				**3**
		Client C	**20**	**8**	**40**		**Yes**			
		Client D	**14**	**6**	**42.85714**		**Yes**			
		Client E	**10**	**3**	**30**	**Yes**				
						upto 10%	**upto 20%**	**upto 30%**	**upto 40%**	
7	**Davachi**	Client A	**10**	**3**	**30**	Yes				
		Client B	**15**	**5**	**33.33333**	**Yes**				**2**
		Client C	**12**	**4**	**33.33333**	**Yes**				
		Client D	**15**	**7**	**46.66667**			**Yes**		
		Client E	**8**	**4**	**50**				**Yes**	
						upto 10%	**upto 20%**	**upto 30%**	**upto 40%**	**3**

8	**CLSA**	Client A	15	6	40		Yes			
		Client B	**18**	**6**	**33.33**	**Yes**				
		Client C	**24**	**12**	**50**				**Yes**	
		Client D	**15**	**4**	**26.66667**	**Yes**				
		Client E	**12**	**5**	**41.66667**			**Yes**		
						upto 10%	**upto 20%**	**upto 30%**	**upto 40%**	
9	**Zensar**	Client A	18	6	33.33333	Yes				
		Client B	**12**	**5**	**41.67**		**Yes**			**3**
		Client C	**24**	**11**	**45.83333**			**Yes**		
		Client D	**10**	**4**	**40**		**Yes**			
		Client E	**15**	**4**	**26.66667**	**Yes**				
						upto 10%	**upto 20%**	**upto 30%**	**upto 40%**	
10	**TechHighway**	Client A	10	4	40		Yes			
		Client B	**15**	**5**	**33.33**	**Yes**				**2**
		Client C	**24**	**10**	**41.66667**		**Yes**			
		Client D	**12**	**4**	**33.33333**	**Yes**				
		Client E	**8**	**3**	**37.5**	**Yes**				

Table 4.29 Software Development Life Cycle during Year 2015-2016

4.9- Testing of Hypotheses

In this research three hypotheses were stated, these entire three hypotheses are tested using SPSS statistics 20 tool, and applied test.

Hypothesis 1-: Z statistics test

Hypothesis 2-: Z statistics test

Hypothesis 3-: Z statistics test for mean test

Hypothesis 1: There are hurdles in collecting SOP process for software development

Collecting SOP Techniques	**Yes**	**No**
Personally Meeting	369	31
Through Document	289	111
Online-Automated	342	58
Lack of Knowledge about the business context	393	7
Lack of Understanding of Business problems/opportunities	400	0
Missing of gaps to be bridged	400	0
Inadequate number of Resources	366	34
Inadequate Time	366	34

Table 4.30 Factors for Collecting SOP from Client

H0 -: 90 % of employees are agreed that there are hurdles in collecting SOP from client.

H1-: more than 90 % of employees are agreed that there are hurdles in collecting SOP from client.

$S.E=\sqrt{PQ/n}$ Where P =0.9

Q = 1-.9 = 0.1

S.E. = $\sqrt{}$ (0.9*0.1 / 3200) = 0.005303

Step IV: Calculation of Z value.

p = Proportion of agreed people = (No. of agreed people / total people)

p = 0.9140625

Z= diff. / S.E.

diff = 0.9140625 – 0.9 = 0.0140625

Zcal = 2.651650429

Step V: Comparison:

Table value of Z for one tail test at 5% level of significance is 1.64

Step VI: Conclusion:

Calculated value of Zcal (i.e. 2.65) > Table value of Z (1.64) **Hence we accept H1** which means more than 90 percent employees are agreed that there are hurdles in collecting SOP from client during software development process and hence the alternate hypothesis "**There are hurdles in collecting SOP process for software development** "of the study is accepted**.**

Hypothesis 2: **IT Industry follows standard practices to use licensed or well-known tools to collect initial SOP from customer in software development.**

Collecting SOP Tools	**Yes**	**No**
Visual Paradigms	336	64
Project Management Software	238	167
Microsoft-Package	235	165

Data Dictionary	166	234
Use Cases and User Stories	160	240
ReqHarbor.com	357	43
MindTool	221	179
IBM Rational Doors	379	21
Jira	146	254
Rally	124	276
Taleo	150	250
Quality Center	318	82

Collecting SOP Tools (The above Table is with reference to same chapter Table no. 4.17).

Step 1: Setting Hypothesis

H0: 95% or more employees agreed that it is best practice to use collecting SOP tools used in IT Industry. (H0: p = .95)

H1: < 95% or more employees agreed that it is best practice to use collecting SOP tools used in IT Industry. (H1= p < .95)

H0 : p =0.95

H1= p < 0.95 (One tail test as rejection area is towards one side)

Step II: Sample Size

n=400 (> 30) As n > 30, large sample test i.e. Z-test is used.

Step III: Calculation of S.E. (Standard Error)

$S.E=\sqrt{pq/n}$ Where p =. 95

q = 1-p =0.0 5

$S.E. = \sqrt{0.95*0.05} / 400 = 0.0108975$

Step IV: Calculation of Z value.

Z= diff. / S.E. diff = 0.9475-0.9500

Zcal= 0.2294

Step V: Comparison:

Table value of Z for one tail test at 5% level of significance is 1.64

Step VI: Conclusion:

Calculated value of Z (0.2294) < Table value of Z (1.64) **Hence we accept H0** which means 95 percent System Analyst have a positive attitude towards **usage of tools for Collecting SOP in IT Industry** and hence the hypothesis "**IT Industry follows standard practices to use licensed or well-known tools to collect initial SOP from customer in software development.**" of the study is accepted**.**

Hypothesis 3: **If collected needs are not freezed, then it has impact on business.**

Referring above table 4.29 following hypothesis is proved

H0 -: On an average, clients are taking 35% of time duration for SOP (i.e $\mu = 0.35$)

H1-: On an average, clients are taking more than 35% of time duration for SOP, which has an impact on business (i.e $\mu > 0.35$)

Step IV: Calculation of Z value.

Sample mean = 0.3893

Population mean under H0 is $\mu = 0.35$

Z= diff. / S.E.

where,

diff = 0.3893 – 0.35 = 0.0393

S.E. = $\sigma/\sqrt{n}$ = 0.009202

Zcal = 0.0393 / 0.009202 = 4.2738

Table value of Z for one tail test at 5% level of significance is Ztab = 1.64

Step VI: Conclusion:

Since Calculated value of Z (4.2738) > Table value of Z (1.64)Hence, we accept H1 which means that, on an average, clients are taking more than 35% of time duration for SOP, which has an impact on business and hence alternate hypothesis " **If collected needs are not freezed, then it has impact on business"** is accepted.

Reference:

1. Vidya Gaveakr, (2013) "A Study of Geographic Information System based computerized framework to enhance the water supply system in Pune City" Thesis of Computer Management Dept., Tilak Maharashtra Univerisity.

2. http://www.enterprisecioforum.com/en/blogs/pearl/it-project-failure-symptoms-and-reasons.
3. K.K. Aggarwal and Yogesh Singh,"Software Engineering",New age International Publishers, third Edition, 2008.
4. http://www.umsl.edu/~sauterv/analysis/Fall2010Papers/Isserman/
5. http://www.practicalecommerce.com
6. http://www.reqharbor.com/
7. https://www.google.co.in/webhp?hl=en#hl=en&q=data%20dictionary
8. https://ankitmathur111.wordpress.com/2012/06/20/whats-whys-hows-of-software-testing-wwh/
9. http://istqbexamcertification.com/what-is-fundamental-test-process-in-software-testing/
10. http://www..ibm.com/software/products/en/ratidoorfami
11. https://scitools.com
12. https://en.wikipedia.org/wiki/Software_development_effort_estimation
13. http://www.jamasoftware.com/blog/change-impact-analysis-2
14. Missing Requirements Information and its Impact on Software Architectures: A Case Study by Md Rounok Salehin at the University of Western Ontario.

Chapter 5

Conclusions and Suggestions

5.1 Conclusions

In the chapter 4 , researcher has completed analysis on data collected through questionnaire, and prepared total 29 analysis tables through which objectives and hypothesis has been proved.

Following are the conclusions made through research.

1. Personally meeting with clients, through documents, online or automated are equally important techniques for collecting unambiguous needs from customer in software development process.
2. Type of customer's needs i.e Using Scope Clarification for Domain, Input Processes, Reporting Procedures, Number of Users and Data Collection are important information which results into error free software product.
3. Time for collecting customer needs required for product development must be Adequate so that proper needs can be collected to avoid further problems on product.
4. "Power up communication with visuals" is useful and effective technique for collecting customer needs through this technique proper set of needs can be gathered.

5. Many people like senior management team, senior architecture team, testers, developers, clients, end users and subscribers are involved in the discussion of collecting needs.
6. Time is the most important factor for software development process because as time increases, schedule may get lagged due to which it may have indirect effect on cost and business.
7. 'Lack of knowledge about the business context' is the most responsible factor for failure of collecting needs from customer process.
8. 'Requirement Errors' is the most responsible factor to make software product erroneous because as we know that requirement is the base for all the phases of SDLC process.
9. 'Lack of user involvement' followed by 'Poor or No Requirements' is most important factor responsible for the failure of software project.
10. 'IBM Rational Doors' and ReqHarbor.com are the best collecting needs tool through which needs are stored in automated format and can be accessed by team of software product development.
11. Automated Testing is the most useful method of software testing in software companies.
12. Execution of 8 testcases per day using automated testing mode is best practice for software testing process to avoid failure of software product.
13. Tester should expect 5 to 6 defects per day in their software testing process to avoid software failure.

14. "Addition of test cases" is the most frequent task tester needs to do when customer's needs are incomplete and changing though its development process.
15. 'Gap in testing or discontinuity in testing work' affects total cost of software project. Cost is indirectly related with schedule of development.
16. Absence and Incompleteness, Incorrectness, Ambiguity and Vagueness, Volatility, Traceability parameters are the most important causes for failure of software.
17. Maximum companies who spent more time on collecting customer's needs incur more cost as cost is indirectly related to time for product development.

18. **Thus the final conclusion is noticed through the research. The process of collecting accurate needs should be well documented to resolve ambiguity because it directly impacts on business of software development. The process of collecting needs must be automated through tools so that ambiguity gets resolved and proper development process will get executed. Once needs gets freezed there should not be delay in schedule for development and thus extra cost should not be incurred for whole software development process.**

5.2 Suggestions

1. It is suggested that developers, software testers should read and understand customer's needs carefully before starting their work.
2. Management of any software company should have adequate number of resources, work allocation between resources should be balanced and requirement engineer should also not spend more time collecting needs from customer.
3. Collected needs must be accurate and well documented.
4. Customer needs must be collected through automated tools.
5. Researcher has suggested a format of documents for collecting customer needs in the proposed model.
6. Software testers should write testcases for accurate customer's needs.
7. Software tester should get involved in the requirement phase and also communicate with requirement engineers for better understanding of customer requirements.
8. Software tester should create correct test cases and test data before starting software testing.
9. Software tester should verify test results with exact functionality required by customers and also consider performance of software system.
10. Software companies should consider all the factors which are responsible for failure and rectify the same immediately.
11. Implementation of model improves the interaction between developer and tester and helps to increase quality of the product.

12. Encourage the software companies for active participation in quality product development and implementation of model in minimal cost.
13. Interaction of researcher from Industry and academia is also required to make constant improvement for successful implementation of model for better quality of product.
14. Conduction of quality audit from third party
15. Software Testing Clubs participation in execution of quality audits with standardized (ISO, CMM, Six Sigma etc) companies.
16. Awareness about the quality standards among the employees of the software companies can be created.
17. To make the employees more productive, thrust on awareness about tools by arranging various training sessions for employees.
18. Make employees aware of their responsibility towards development of quality product.
19. There should be QA team activity on feedback system for employees on quality development, tester performance improvement.
20. Organize quality product fest program to create awareness about quality product among the employees.
21. QA team should organize award and recognition fest for successfully and error free development of software.
22. Active involvement of finance manager throughout SDLC will help in keeping track of the cost.

23. The manuals must be provided and followed by employees during implementation of model to avoid errors.
24. There should be up gradation of tools used for development and testing.
25. Organization should purchase upgraded version and licensed of various tools used in software companies.
26. By using tools organization saves resources like time, efforts, and cost.
27. By using automated reverse engineering tool requirement changes can be easily traced out in the development process.
28. In suggested model, input model stores all customer's needs document category wise like missing requirement, wrong requirement etc. which will help to avoid errors in the product.
29. Timely freezing of customer's needs will lead to better utilization of resources like cost and time.

5.3 Suggested Model

Customer's Needs Management to Reduce Software Failures Model (CNMRSF)

Considering the present state of collecting needs from customer process for software development process, a model to reduce software failure in testing phase is designed through the present research work. This new model is called Customer's Needs Management to Reduce Software Failures (CNMRSF). The main functionality of CNMRSF model is to provide better software testing actions for corresponding poor requirement. Model always has 3 phases like input, processing and output [1, 2]. CNMRSF model integrates the functionality of different modules like Input module, processing module and output module. Figure 5.1 shows the work flow of CNMRSF

model. The workflow of CNMRSF model is divided in 3 phases: Input phase, processing phase and Output generation phase.

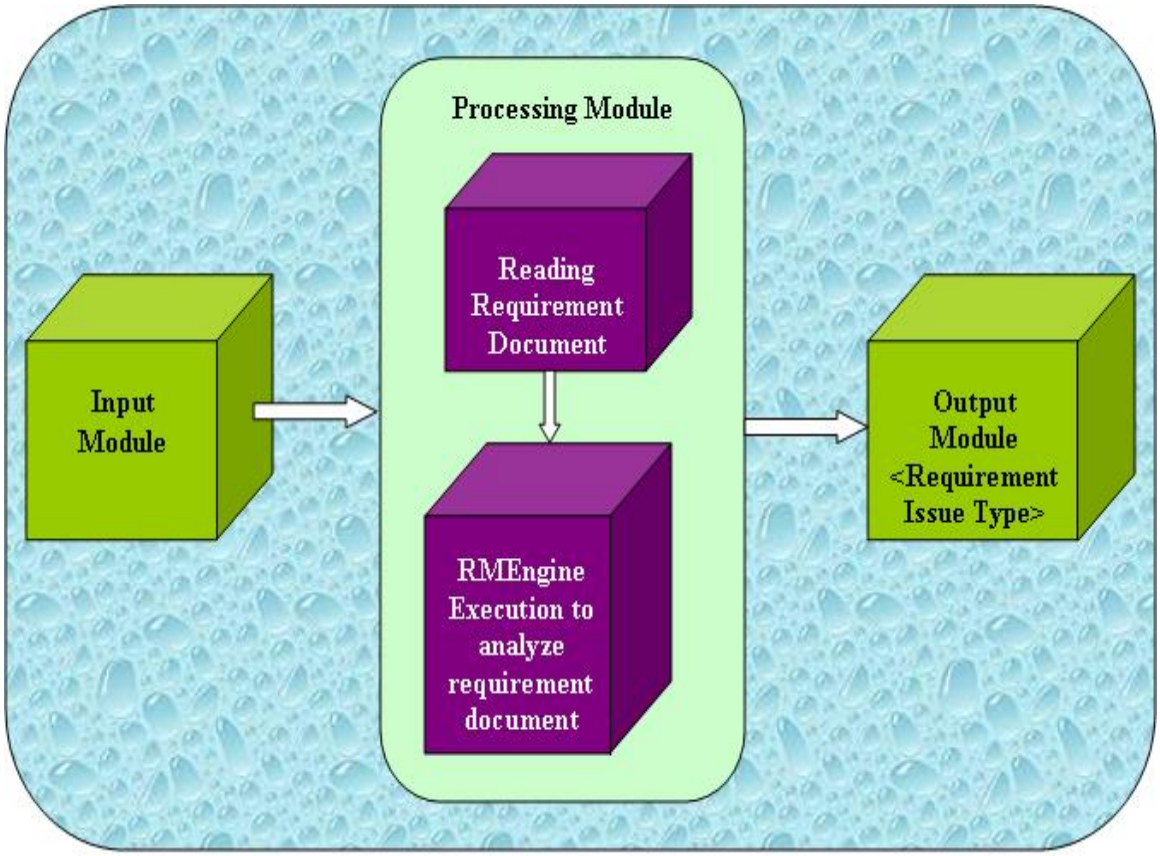


Figure 5.1 High level architecture of CNMRSF Model

In **Input Module** Customer Needs document can get from your local computer drive. Main functionality of this phase is to get exact type and requirement document for further analysis.

Processing Module deals with integrated functionality of Reading Requirement document, analyzing requirement document by call customer's Needs Engine (RA Engine) and Execution of Output module to generate list of Issues, Impacts and Actions for particular type of requirement issue.

Output Module deals with generation of output based on input argument as a requirement issue type provided by processing phase. Output phase has module named as Output Module and this module is basically gets executed by Processing module. Main functionality of output module is to get requirement type issue as an input and based on this input query to database to fetch corresponding list of Issues, Impacts and Action points. This module displaying list of Issues, Impacts and Actions based on corresponding requirement issue type.

Above Model is described in detail in Annexure -II

Customer Needs Management to Reduce Software Failure model has been designed to analyze the provided requirement document for the recognition of exact requirement issue. And based on requirement issue it is providing list of requirement issues, impact of this requirement issue and actions need to be performed by testing team to reduce extra efforts and failures in software product.

References

[1]	Thomas Kiihne , "What is Model?" Darmstadt University of Technology, Darmstadt, Germany.
[2]	http://www.la1.psu.edu/cas/jgastil/pdfs/conceptualdefinitiondeliberation.pdf
[3]	http://www.voltreach.com/Development_Methodologies.aspx
[4]	Kirsten Kiefer, "The Impact of Requirement issues on testing", Software

	Education associates Ltd.
[5]	http://www.softed.com/resources/docs/impact-of-requirements-issues-on-testing.pdf
[6]	People soft upgrade process for testing.
[7]	http://blog.sei.cmu.edu/post.cfm/common-testing-problems-pitfalls-to-prevent-and-mitigate

ANNEXURE-I

A Study of Collecting Customer Needs in Software Development Process and Its Impact on Business of Selected IT Companies in Pune.

NOTE: In today's IT world, more and more online Applications and tools are used, which help in carrying out various daily chores. If these applications and tools do not work according to specification then it would cause inconvenience to all users. The researcher is an academician who is interested in surveying the causes of failures of softwares. For a better feedback from the Industry Experts from Pune IT Hub, I would appreciate your cooperation in responding to this questionnaire.

1 Company(Optional) _____

2. Gender : a) Male b) Female

3. Age : a)Upto 40 b) 41-50 c) 51-60 d) Above 60

4. Education a) Graduate b) Post Graduate

c) Any Other (Specify)_____

5. Occupation: a) Business Analyst b) Designer c) Tester

d) Any Other :_____

6 OfficeLocation a) Hadapsar b) Hinjewadi c) Kharadi

d) Any Other

7 From whom you collect requirements (Tick Multiple Option)

- Stakeholders
- End User

8 How you collect requirements from clients.(Tick Multiple Option)

a) Personally meetings with Clients

b) Through Documents

c) Online or Automated

d) Any other:_____

9 which kind of requirements you collect.(Tick Multiple Option)

a) Scope Clarification for Domain

b) Input Processes

c) Reporting Procedure

d) Number of users

e) Data Collection

f) All of the above

g) Any other:_____

10 who involved in requirement analysis process

a. Senior Management
b. Project Manager
c. End User
d. Requirement Team

e. Developers
f. Testers

g. Scribes

h. Any other:_____

11. How much times you interact or communicate with clients during requirements gathering process.

a. 2 Weeks
b. 3 Weeks
c. 4 Weeks
d. More than weeks
e. Any Other: _____

12 For single interaction, how much time required

a. Less than 2 Hrs.
b. 3 Hrs.
c. 4 Hrs.
d. More than 4Hrs.
e. Any Other: _____

13 Are you aware of following Software Development Life Cycle process?

Sr, No	Model Name	Yes		No
1	Waterfall			
2	Rapid Prototyping			
3	Incremental			
4	Agile			
5	Spiral			

14 Which of the following technique is most beneficial to gather software requirement and to what extent? (Tick any one for Strongly Agree (SA)-5, Agree(A)-4, Neutral(N)-3, Disagree(D)-2, Strongly Disagree(DS)-1)

Sr.No.	Techniques	SA(5)	A(4)	N(3)	D(2)	SD(1)
a)	Homework Completion					
b)	Power up Communication with					
	Visuals					
C)	Use of standard template to support					
	your work					
d)	Avoid common pitfalls					
e)	Use of tools					

15 State the significance of following documents in success of software project? Strongly Agree (SA)-5, Agree(A)-4, Neutral(N)-3, Disagree(D)-2, Strongly Disagree(DS)-1

Sr.No	Factors	SA(5)	A(4)	N(3)	D(2)	SD(1)
a)	Customer Requirement Document					
	(CRD)					
b)	Business Requirement Document					
	(BRD)					
c)	Functional Requirement Document					
	(FRD)					
d)	Component Specification Document					
	(CSD)					
e)	Component Design Document					
	(CDD)					
f)	Test cases Document (TCD)					

16 What do you think which factor of the following is reasonable for the failure of software project?

Strongly Agree (SA)-5, Agree(A)-4, Neutral(N)-3, Disagree(D)-2, Strongly Disagree(DS)-1

Sr.No	**Factors**	**SA(5)**	**A(4)**	**N(3)**	**D(2)**	**SD(1)**
a)	Lack of user involvement					
b)	Long or unrealistic time scale					
c)	Poor or No Requirements					
d)	Inadequate Documentations					
e)	Scope Creep					
f)	No Change Control System					
g)	Poor testing					
h)	Lack of foresight in building efficiency					
	markets					
i)	poor managerial decisions					
j)	Cost overrun.					
k)	Lack of an experienced project					
	manager:					
l)	Lack of methodology in the process					
m)	Well-defined Schedules					

17 What are the factors contributing to failure forrequirement gathering process Give in terms of weightage (1-5)?

Sr.No	Factors	SA(5)	A(4)	N(3)	D(2)	SD(1)
a)	Knowledge about the business					
	context					
b)	Understanding of Business					
	problems/opportunities					
c)	Identification of gaps to be bridged					
d)	Adequate number of Resources					
e)	Adequate Time					

18 According to your opinion from the beginning of SDLC to what extent testers plays a role. Option)

Role	100% - 90%	90%-70%	70% - 50%	Less than 50%
Tester				

19 Whether Tester is present In all the below Activities of Project?

Sr.No	Factors	Yes	No
a)	Requirement Phase		
b)	Design Phase		
c)	Development Phase		
d)	Testing Phase		
e)	Maintenance Phase		

20 How non-requirement gathering time is gettingspent in yourorganization please mention in Percentage?

Sr.No	Factors	Enter in the Percentage
a)	Writing Requirement Documents	
b)	Reviewing FRD/BRD	
c)	Client Customer interaction	
d)	Conducting trainings for Tester and	
	Developers	

e)	Any Others	

21 If yes then please mention which tools are getting used to gather requirements in your organization?

Sr.No	**Factors**	Tick
a)	Visual Paradigms	
b)	Project Management Software	
c)	MicrosoftPackage	
d)	Data Dictionary (Service Versioning Number [SVN])	
e)	Use Cases and User Stories	
f)	ReqHarbor.com	
g)	MindTool	
h)	IBM Rational Doors	
i)	Jira	
j)	Rally	
k)	Taleo	
l)	Quality Center	

22 Are you using testing tools?

- Yes
- No

23 Which testing type are your preferring?

- Automated Testing
- Manual Testing

24 Which of the following tools are you using for software testing?

i) Soap Box test tool

ii) QTP

iii) jmeter

iv) Load Tracer

v) Specify here if any other ……

25 How many test cases are you executing per day?

- 4

- 8
- 16
- 20

26 Per day, how many defects are getting raised on incorrect requirement?

- 1-2
- 3-4
- 5-6
- 7-8

27 Following are the different cost that are considered during testing process. Give your opinion in terms of weightage (1-5) regarding the level of losses occurred as per this cost?

Sr. No.	**Factors (utilization)**	**SA(5)**	**A(4)**	**N(3)**	**D(2)**	**SD(1)**
a)	Resources					
b)	Software					
c)	Hardware					
d)	Network					
e)	Infrastructure(electricity, rent etc)					
f)	Any other					

28 State the following document is most useful in success of software testing?

Strongly Agree (SA)-5, Agree(A)-4, Neutral(N)-3, Disagree(D)-2, Strongly Disagree(DS)-1

Sr.No.	**Factors**	**SA(5)**	**A(4)**	**N(3)**	**D(2)**	**SD(1)**
a)	Customer Requirement Document					
	(CRD)					
b)	Business Requirement Document					
	(BRD)					
c)	Functional Requirement Document					
	(FRD)					
d)	Component Specification Document					
	(CSD)					
e)	Component Design Document					
	(CDD)					
f)	Test cases Document (TCD)					

29 State how poor requirement gathering or change in requirement gathering may impact on software testing process?

Strongly Agree (SA)-5, Agree(A)-4, Neutral(N)-3, Disagree(D)-2, Strongly Disagree(DS)-1

Sr.No.	Factors	SA(5)	A(4)	N(3)	D(2)	SD(1)
a)	Addition of Test case					
b)	Deletion of Test case					
c)	Modification of existing test case					
d)	Re-execution of modified test case					
e)	Verification of newly added					
f)	functionality due to requirement Change					
g)	Test result creation for newly added					
	Requirement					

30 State following which factors is responsible to make software erroneous.

Strongly Agree (SA)-5, Agree(A)-4, Neutral(N)-3, Disagree(D)-2, Strongly Disagree(DS)-1

Sr.No	Factors	SA(5)	A(4)	N(3)	D(2)	SD(1)
a)	Logic Design					
b)	Documentation					
c)	Human					
d)	Environment					
e)	Data					
f)	Interface					
g)	Requirement Errors					
f)	Any other					

31 State overhead occurs in software testing due to poor requirement gathering.

Strongly Agree (SA)-5, Agree(A)-4, Neutral(N)-3, Disagree(D)-2, Strongly Disagree(DS)-1

Sr.No	Factors	SA(5)	A(4)	N(3)	D(2)	SD(1)
a)	Gap in Testing					
b)	Increase in system failures					
c)	System Testing Delay					
d)	Inaccurate Testing Estimation					
e)	Test_team_credibility_decreation					
f)	Delay_benefit_realisation					

32 What are the common requirement issues that may affect Software Testing?

Strongly Agree (SA)-5, Agree(A)-4, Neutral(N)-3, Disagree(D)-2, Strongly Disagree(DS)-1

Sr.No	Factors	SA(5)	A(4)	N(3)	D(2)	SD(1)
a)	Absence and Incompleteness					
b)	Incorrectness					
c)	Ambiguity and Vagueness					
d)	Volatility					
e)	Traceability					

33 Following efforts are carried out in case of collected needs from customer are not freeze.

Sr.No.	Tool Name	Tick
1	Development Efforts	
2	Rework Efforts	
3	Quality Assurance Efforts	
4	Testing Efforts	

The information given by the respondent would be treated as confidential.

Annexure-II

Conceptual Background of Software Requirement Analysis and Software Testing Process

A requirement is an expression of desired behaviour. A requirement is nothing but objects or entities, the states they can be in, and the functions that are performed to change states or object characteristics. [1] The goal of the requirements phase is premature until the problem is clearly defined [1] to understand the customer's problems and needs. Thus, requirements focus on the customer and the problem, not on the solution or the implementation. We often say that requirements designate what behaviour the customer wants, without saying how that behaviour will be realized. This chapter mainly focuses on detail understanding of requirement engineering process, software testing process and how poor requirement gathering process impacts on software testing process.

Requirement Engineering

The process to gather the software requirements from client, analyse and document them is known as requirement engineering. The goal of requirement engineering is to develop and maintain sophisticated and descriptive 'System Requirements Specification' document.

Requirement Engineering Process

It is a four-step process, which includes –

(1) Feasibility Study

(2) Requirement Gathering

(3) Software Requirement Specification

(4) Software Requirement Validation

Feasibility study

When the client approaches the organization for getting the desired product developed, it comes up with rough idea about what all functions the software must perform and which all features are expected from the software. Referencing to this information, the analysts does a detailed study about whether the desired system and its functionality are feasible to develop. This feasibility study is focused towards goal of the organization. This study analyses whether the software product can be practically materialized in terms of implementation, contribution of project to organization, cost constraints and as per values and objectives of the organization. It explores technical aspects of the project and product such as usability, maintainability, and productivity and integration ability. The output of this phase should be a feasibility study report that should contain adequate comments and recommendations for management about whether or not the project should be undertaken.

Requirement Gathering

If the feasibility report is positive towards undertaking the project, next phase starts with gathering requirements from the user. Analysts and engineers communicate with the client and end-users to know their ideas on what the software should provide and which features they want the software to include.

Software Requirement Specification

SRS is a document created by system analyst after the requirements are collected from various stakeholders. SRS defines how the intended software will interact with hardware, external

interfaes, speed of operation, response time of system, portability of software across various platforms, maintainability, speed of recovery after crashing, Security, Quality, Limitations etc. The requirements received from client are written in natural language. It is the responsibility of system analyst to document the requirements in technical language so that they can be comprehended and useful by the software development team. SRS should come up with following features:

- User Requirements are expressed in natural language.
- Technical requirements are expressed in structured language, which is used inside the organization.
- Design description should be written in Pseudo code.
- Format of Forms and GUI screen prints.
- Conditional and mathematical notations for DFDs etc.

Software Requirement Validation

After requirement specifications are developed, the requirements mentioned in this document are validated. User might ask for illegal, impractical solution or experts may interpret the requirements incorrectly. This results in huge increase in cost if not nipped in the bud. Requirements can be checked against following conditions –

- If they can be practically implemented
- If they are valid and as per functionality and domain of software
- If there are any ambiguities
- If they are complete
- If they can be demonstrated

Annexure -II

Requirement Gathering Process

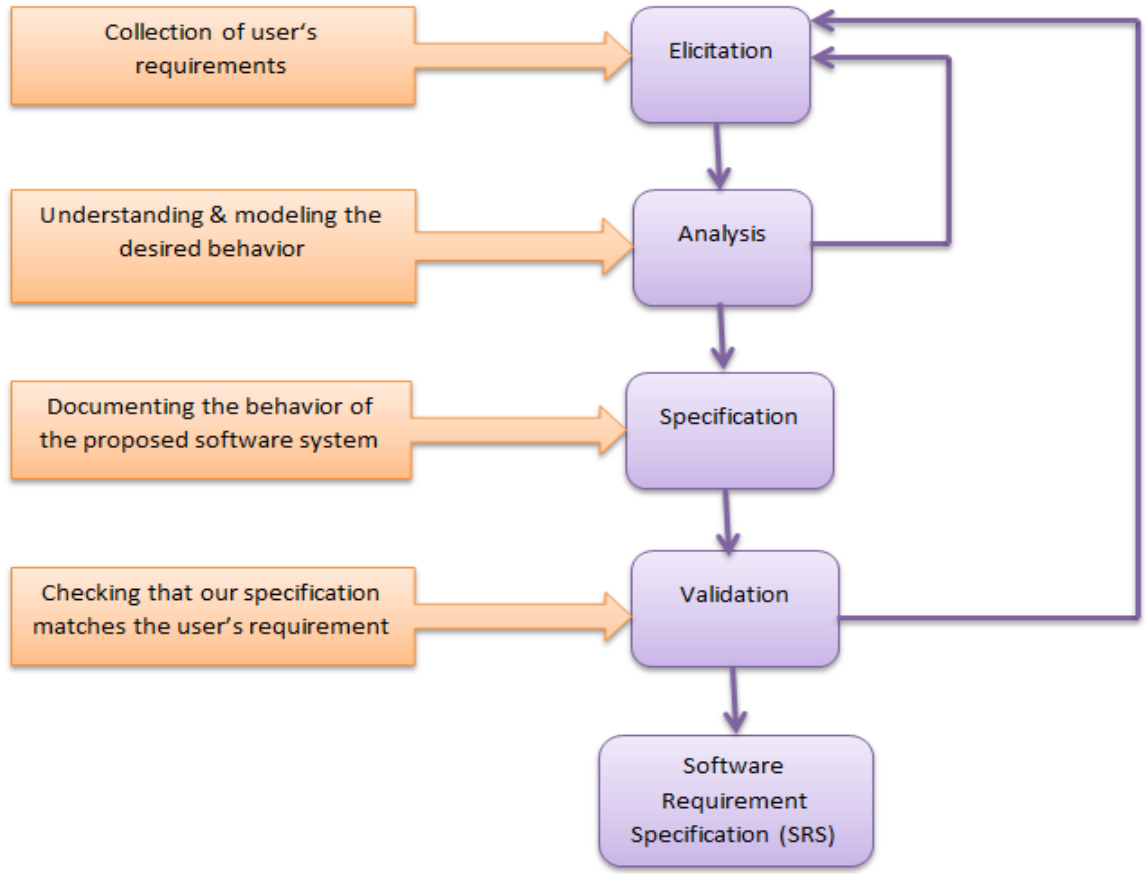


Figure 1 Requirement Gathering Process

Figure 1. Illustrates the process of determining the requirements for a proposed software system. The person performing these tasks usually goes by the title of requirements analyst or systems analyst. As a requirements analyst, we first work with our customers to elicit the requirements, by asking questions, examining current behavior, or demonstrating similar systems. Next, we capture the requirements in a model or a prototype. This exercise helps business analyst to better understand the required behavior, and usually raises additional

questions about what the customer wants to happen in certain situations. Once the requirements are well understood, we progress to the specification phase, in which business analyst decide which parts of the required behavior will be implemented in software. During validation, business analyst checks that requirement specification matches what the customer expects to see in the final product. Analysis and validation activities may expose problems or omissions in the models or specification that cause us to revisit the customer and revise our models and specification. The eventual outcome of the requirements process is a Software Requirements Specification (SRS), which is used to communicate to other software developers (designers, testers, maintainers) how the final product ought to behave.

Software Testing

Testing is the process of evaluating a system or its component(s) with the intent to find whether it satisfies the specified requirements or not.

Testing is executing a system in order to identify any gaps, errors, or missing requirements in contrary to the actual requirements.

Testing is the process of evaluating a system or its component(s) with the intent to find whether it satisfies the specified requirements or not. In simple words, testing is executing a system in order to identify any gaps, errors, or missing requirements in contrary to the actual requirements.

According to ANSI/IEEE 1059 standard, Testing can be defined as - A process of analyzing a software item to detect the differences between existing and required conditions (that is defects/errors/bugs) and to evaluate the features of the software item.

Annexure -II

Main functionality of Software Testing [3]

Main focus of Software testing functionality basically deals with following four aspects:

Verification

Verification addresses the concern: "Are you building it right?" Ensures that the software system meets all the functionality. Verification takes place first and includes the checking for documentation, code, etc. It has static activities, as it includes collecting reviews, walkthroughs, and inspections to verify a software. It is an objective process and no subjective decision should be needed to verify a software. Done by developers.

Validation

Validation addresses the concern: "Are you building the right thing?" Ensures that the functionalities meet the intended behavior. Validation occurs after verification and mainly involves the checking of the overall product. Done by testers. It has dynamic activities, as it includes executing the software against the requirements. It is a subjective process and involves subjective decisions on how well a software works.

Testing

It involves identifying bug/error/defect in a software without correcting it. Normally professionals with a quality assurance background are involved in bugs identification. Testing is performed in the testing phase.

Debugging

It involves identifying, isolating, and fixing the problems/bugs. Developers who code the software conduct debugging upon encountering an error in the code. Debugging is a part of White Box Testing or Unit Testing. Debugging can be performed in the development phase while conducting Unit Testing or in phases while fixing the reported bugs.

Annexure -II

Types of Software Testing

There is different way to test any software product. Following are the few testing types are studied in this research.

Manual Testing

Manual testing includes testing a software manually, i.e., without using any automated tool or any script. In this type, the tester takes over the role of an end-user and tests the software to identify any unexpected behavior or bug. There are different stages for manual testing such as unit testing, integration testing, system testing, and user acceptance testing.

Testers use test plans, test cases, or test scenarios to test a software to ensure the completeness of testing. Manual testing also includes exploratory testing, as testers explore the software to identify errors in it.

Automation Testing

Automation testing, which is also known as Test Automation, is when the tester writes scripts and uses another software to test the product. This process involves automation of a manual process. Automation Testing is used to re-run the test scenarios that were performed manually, quickly, and repeatedly.

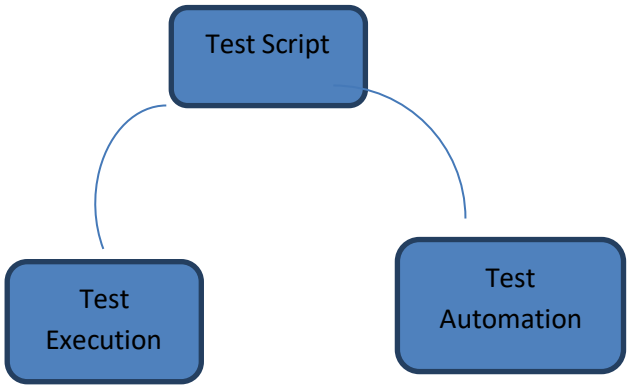


Figure 2 Automation Testing

Annexure -II

Apart from regression testing, automation testing is also used to test the application from load, performance, and stress point of view. It increases the test coverage, improves accuracy, and saves time and money in comparison to manual testing.

Black-Box Testing

The technique of testing without having any knowledge of the interior workings of the application is called black-box testing. The tester is oblivious to the system architecture and does not have access to the source code. Typically, while performing a black-box test, a tester will interact with the system's user interface by providing inputs and examining outputs without knowing how and where the inputs are worked upon.

White-Box Testing

White-box testing is the detailed investigation of internal logic and structure of the code. White-box testing is also called glass testing or open-box testing. In order to performwhite-box testing on an application, a tester needs to know the internal workings of the code.

The tester needs to have a look inside the source code and find out which unit/chunk of the code is behaving inappropriately.

Grey-Box Testing

Grey-box testing is a technique to test the application with having a limited knowledge of the internal workings of an application. In software testing, the phrase the more you know, the better carries a lot of weight while testing an application.

Mastering the domain of a system always gives the tester an edge over someone with limited domain knowledge. Unlike black-box testing, where the tester only tests the application's user interface; in grey-box testing, the tester has access to design documents and the database.

Having this knowledge, a tester can prepare better test data and test scenarios while making a test plan.

There are different levels during the process of testing. In this chapter, a brief description is provided about these levels.

Levels of testing include different methodologies that can be used while conducting software testing. The main levels of software testing are:

- Functional Testing
- Non-functional Testing

Functional Testing

This is a type of black-box testing that is based on the specifications of the software that is to be tested. The application is tested by providing input and then the results are examined that need to conform to the functionality it was intended for. Functional testing of software is conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements.

There are five steps that are involved while testing an application for functionality.

- The determination of the functionality that the intended application is meant to perform.
- The creation of test data based on the specifications of the application.
- The output based on the test data and the specifications of the application.
- The writing of test scenarios and the execution of test cases.
- The comparison of actual and expected results based on the executed test cases.

Annexure -II

An effective testing practice will see the above steps applied to the testing policies of every organization and hence it will make sure that the organization maintains the strictest of standards when it comes to software quality.

Unit Testing

This type of testing is performed by developers before the setup is handed over to the testing team to formally execute the test cases. Unit testing is performed by the respective developers on the individual units of source code assigned areas. The developers use test data that is different from the test data of the quality assurance team.

The goal of unit testing is to isolate each part of the program and show that individual parts are correct in terms of requirements and functionality.

Integration Testing

Integration testing is defined as the testing of combined parts of an application to determine if they function correctly. Integration testing can be done in two ways: Bottom-up integration testing and Top-down integration testing.

Integration Testing Methods:

Bottom-up integration

This testing begins with unit testing, followed by tests of progressively higher-level combinations of units called modules or builds.

Top-down integration

In this testing, the highest-level modules are tested first and progressively, lower-level modules are tested thereafter.

Annexure -II

In a comprehensive software development environment, bottom-up testing is usually done first, followed by top-down testing. The process concludes with multiple tests of the complete application, preferably in scenarios designed to mimic actual situations.

System Testing

System testing tests the system as a whole. Once all the components are integrated, the application as a whole is tested rigorously to see that it meets the specified Quality Standards. This type of testing is performed by a specialized testing team.

System testing is important because of the following reasons:

- System testing is the first step in the Software Development Life Cycle, where the application is tested as a whole.
- The application is tested thoroughly to verify that it meets the functional and technical specifications.
- The application is tested in an environment that is very close to the production environment where the application will be deployed.
- System testing enables us to test, verify, and validate both the business requirements as well as the application architecture.

Regression Testing

Whenever a change in a software application is made, it is quite possible that other areas within the application have been affected by this change. Regression testing is performed to verify that a fixed bug hasn't resulted in another functionality or business rule violation. The intent of regression testing is to ensure that a change, such as a bug fix should not result in another fault being uncovered in the application.

Regression testing is important because of the following reasons:

- Minimize the gaps in testing when an application with changes made has to be tested.
- Testing the new changes to verify that the changes made did not affect any other area of the application.
- Mitigates risks when regression testing is performed on the application.
- Test coverage is increased without compromising timelines.
- Increase speed to market the product.

Acceptance Testing

This is arguably the most important type of testing, as it is conducted by the Quality Assurance Team who will gauge whether the application meets the intended specifications and satisfies the client's requirement. The QA team will have a set of pre-written scenarios and test cases that will be used to test the application.

More ideas will be shared about the application and more tests can be performed on it to gauge its accuracy and the reasons why the project was initiated. Acceptance tests are not only intended to point out simple spelling mistakes, cosmetic errors, or interface gaps, but also to point out any bugs in the application that will result in system crashes or major errors in the application.

By performing acceptance tests on an application, the testing team will deduce how the application will perform in production. There are also legal and contractual requirements for acceptance of the system.

Alpha Testing

This test is the first stage of testing and will be performed amongst the teams (developer and QA teams). Unit testing, integration testing and system testing when combined together is

known as alpha testing. During this phase, the following aspects will be tested in the application:

- Spelling Mistakes
- Broken Links
- Cloudy Directions
- The Application will be tested on machines with the lowest specification to test loading times and any latency problems.

Beta Testing

This test is performed after alpha testing has been successfully performed. In beta testing, a sample of the intended audience tests the application. Beta testing is also known as pre-release testing. Beta test versions of software are ideally distributed to a wide audience on the Web, partly to give the program a "real-world" test and partly to provide a preview of the next release. In this phase, the audience will be testing the following:

- Users will install, run the application and send their feedback to the project team.
- Typographical errors, confusing application flow, and even crashes.
- Getting the feedback, the project team can fix the problems before releasing the software to the actual users.
- The more issues you fix that solve real user problems, the higher the quality of your application will be.
- Having a higher-quality application when you release it to the general public will increase customer satisfaction.

Non-Functional Testing

Annexure -II

This section is based upon testing an application from its non-functional attributes. Non-functional testing involves testing a software from the requirements which are nonfunctional in nature but important such as performance, security, user interface, etc.

Some of the important and commonly used non-functional testing types are discussed below.

Performance Testing

It is mostly used to identify any bottlenecks or performance issues rather than finding bugs in a software. There are different causes that contribute in lowering the performance of a software:

- Network delay
- Client-side processing
- Database transaction processing
- Load balancing between servers
- Data rendering

Performance testing is considered as one of the important and mandatory testing type in terms of the following aspects:

- Speed (i.e. Response Time, data rendering and accessing)
- Capacity
- Stability
- Scalability

Performance testing can be either qualitative or quantitative and can be divided into different sub-types such as Load testing and Stress Testing.

Annexure -II

Load Testing

It is a process of testing the behaviour of a software by applying maximum load in terms of software accessing and manipulating large input data. It can be done at both normal and peak load conditions. This type of testing identifies the maximum capacity of software and its behaviour at peak time.

Most of the time, load testing is performed with the help of automated tools such as Load Runner, AppLoader, IBM Rational Performance Tester, Apache JMeter, Silk Performer, Visual Studio Load Test, etc.

Virtual users (VUsers) are defined in the automated testing tool and the script is executed to verify the load testing for the software. The number of users can be increased or decreased concurrently or incrementally based upon the requirements.

Stress Testing

Stress testing includes testing the behaviour of a software under abnormal conditions. For example, it may include taking away some resources or applying a load beyond the actual load limit.

The aim of stress testing is to test the software by applying the load to the system and taking over the resources used by the software to identify the breaking point. This testing can be performed by testing different scenarios such as:

- Shutdown or restart of network ports randomly
- Turning the database on or off
- Running different processes that consume resources such as CPU, memory, server, etc.

Annexure -II

Usability Testing

Usability testing is a black-box technique and is used to identify any error(s) and improvements in the software by observing the users through their usage and operation.

According to Nielsen, usability can be defined in terms of five factors, i.e. efficiency of use, learn-ability, memory-ability, errors/safety, and satisfaction. According to him, the usability of a product will be good and the system is usable if it possesses the above factors.

Nigel Bevan and Macleod considered that usability is the quality requirement that can be measured as the outcome of interactions with a computer system. This requirement can be fulfilled and the end-user will be satisfied if the intended goals are achieved effectively with the use of proper resources.

Molich in 2000 stated that a user-friendly system should fulfill the following five goals, i.e., easy to Learn, easy to remember, efficient to use, satisfactory to use, and easy to understand.

In addition to the different definitions of usability, there are some standards and quality models and methods that define usability in the form of attributes and sub-attributes such as ISO-9126, ISO-9241-11, ISO-13407, and IEEE std.610.12, etc.

UI vs. Usability Testing

UI testing involves testing the Graphical User Interface of the Software. UI testing ensures that the GUI functions according to the requirements and tested in terms of color, alignment, size, and other properties.

On the other hand, usability testing ensures a good and user-friendly GUI that can be easily handled. UI testing can be considered as a sub-part of usability testing.

Security Testing

Annexure -II

Security testing involves testing software in order to identify any flaws and gaps from security and vulnerability point of view. Listed below are the main aspects that security testing should ensure:

- Confidentiality
- Integrity
- Authentication
- Availability
- Authorization
- Non-repudiation
- Software is secure against known and unknown vulnerabilities
- Software data is secure
- Software is according to all security regulations
- Input checking and validation
- SQL insertion attacks
- Injection flaws
- Session management issues
- Cross-site scripting attacks
- Buffer overflows vulnerabilities
- Directory traversal attacks

Portability Testing

Portability testing includes testing a software with the aim to ensure its reusability and that it can be moved from another software as well. Following are the strategies that can be used for portability testing:

- Transferring an installed software from one computer to another.
- Building executable (.exe) to run the software on different platforms.

Portability testing can be considered as one of the sub-parts of system testing, as this testing type includes overall testing of a software with respect to its usage over different environments. Computer hardware, operating systems, and browsers are the major focus of portability testing. Some of the pre-conditions for portability testing are as follows:

- Software should be designed and coded, keeping in mind the portability requirements.
- Unit testing has been performed on the associated components.
- Integration testing has been performed.
- Test environment has been established.

Testing documentation involves the documentation of artifacts that should be developed before or during the testing of Software.

Documentation for software testing helps in estimating the testing effort required, test coverage, requirement tracking/tracing, etc. This section describes some of the commonly used documented artifacts related to software testing such as:

- Test Plan
- Test Scenario
- Test Case
- Traceability Matrix

Software TestingTools

From the survey carried out in this research, it has been seen that following list of tools are getting used by tester for automation testing.

- HP Quick Test Professional

Annexure -II

- Selenium
- IBM Rational Functional Tester
- SilkTest
- TestComplete
- Testing Anywhere
- WinRunner
- LaodRunner
- Visual Studio Test Professional
- WATIR

Impact of poor requirement on Software testing

As we saw in above Requirement gathering section, requirement collection from client is one of the basic and important task of requirement gathering process. But if due to inadequate knowledge of business or functional requirement, business analyst can collect incorrect requirements from client. collected from client then it is definitely going to impact of software development and software testing process. There are many software product failures examples in the world because of incorrect, incomplete requirements. Literature review has shown the many reasons for IT project failure in all over the world [7]. Out of 100% project success rates were only 34% with the rest of project being either “challenged” in some way or failing outright. The failure in software project means there is loss in productivity, revineo of Software Company and these losses are very significant. For example, British food retailer Sainsbury had to write off its $526 million investment in an automated supply-chain management system. The U.S. Federal Aviation Administration spent $2.6 billion unsuccessfully trying to upgrade its air traffic control system in the 1990s. Ford Motor Company abandoned its purchasing

system in 2004, after spending $400 million. In the 8 years since, things probably haven't changed much.[7]

One of main reason of such project failure is incomplete software requirement which in turn happened due to poor requirement gathering. In SDLC process, most of time it is impossible to have complete and finalized set of requirements at the beginning of a project. This leads requirement changes to happen during the latter stages of the project and create conflicts with the software process been practiced [5].

Actually requirement gathering is the practice of collecting the requirements of a system from users, customers and other stakeholders. [6] Requirements gathering practices include interviews, questionnaires, user observation, workshops, brainstorming, use cases, role playing and prototyping.

One of the root causes of poor requirement gathering in SDLC is the only role for users is in specifying requirements, and that all requirements can be specified in advance. Unfortunately, requirements grow and change throughout the process and beyond, calling for considerable feedback and iterative consultation. Due to this frequently changing requirements gathering process, many developers complain about inadequate, non-freezing requirements and its impact on their work, software productivity and time consuming overhead. This non-freezable requirement gathering process not only affects developer’s work but also affecting Tester, maintenance and management team. Non-Freezable requirement leads to poor software requirement gathering and in turn leads to non-qualitative software product. Poor requirement gathering mostly happens due to business problem, and not a technology problem.

Annexure -II

The non-freezable requirements for software, as delivered by typical business analysts, designer is not sufficiently clear, insightful, or well understood to develop software systems that meet the needs of business users.

To overcome this problem, there is need to understand root cause of poor software requirements gathering process and find out the corrective solution for the same.

- **Few cause that leads to poor requirement gathering [7]:**

1. Poor Requirements Quality
2. Over Emphasis on Simplistic Use Case Modeling
3. Inappropriate Constraints
4. Requirements Not Traced
5. Excessive Requirements Volatility including Unmanaged Scope Creep
6. Inadequate Verification of Requirements Quality
7. Inadequate Requirements Validation
8. Inadequate Requirements Management
9. Inadequate Requirements Process
10. Inadequate Tool Support
11. Unprepared Requirements Engineers

Poor requirement happens due to the problems that indicate the challenges for requirements gathering [8]. Following are few challenges that we need to consider while doing requirement gathering.

- '**Problems of scope**'. The boundary of the system is ill-defined or the customers/users specify unnecessary technical detail that may confuse, rather than clarify, overall system objectives.

- **Problems of understanding**. The customers/users are not completely sure of what is needed, have a poor understanding of the capabilities and limitations of their computing environment, don't have a full understanding of the problem domain, have trouble communicating needs to the system engineer, omit information that is believed to be "**obvious**," specify requirements that conflict with the needs of other customers/users, or specify requirements that are ambiguous or untestable.
- **Problems of volatility**. The requirements change over time. The rate of change is sometimes referred to as the level of requirement volatility

Impact of Poor Requirements on Software Project and Customer's Business

The quality of requirements can have a lot of impact on the outcome of the project. One high profile project which was significantly affected by the requirements management process was the Chrysler Comprehensive Compensation System which was supposed to handle paychecks for Chrysler's 87,000 employees but was shut down after several years of development.

The impact is magnified as the BA moves from high-level requirements towards functional and non-functional requirements. The cost of rework of functional requirements is the highest because these requirements define the technical specification and design of the solution.

Annexure -II

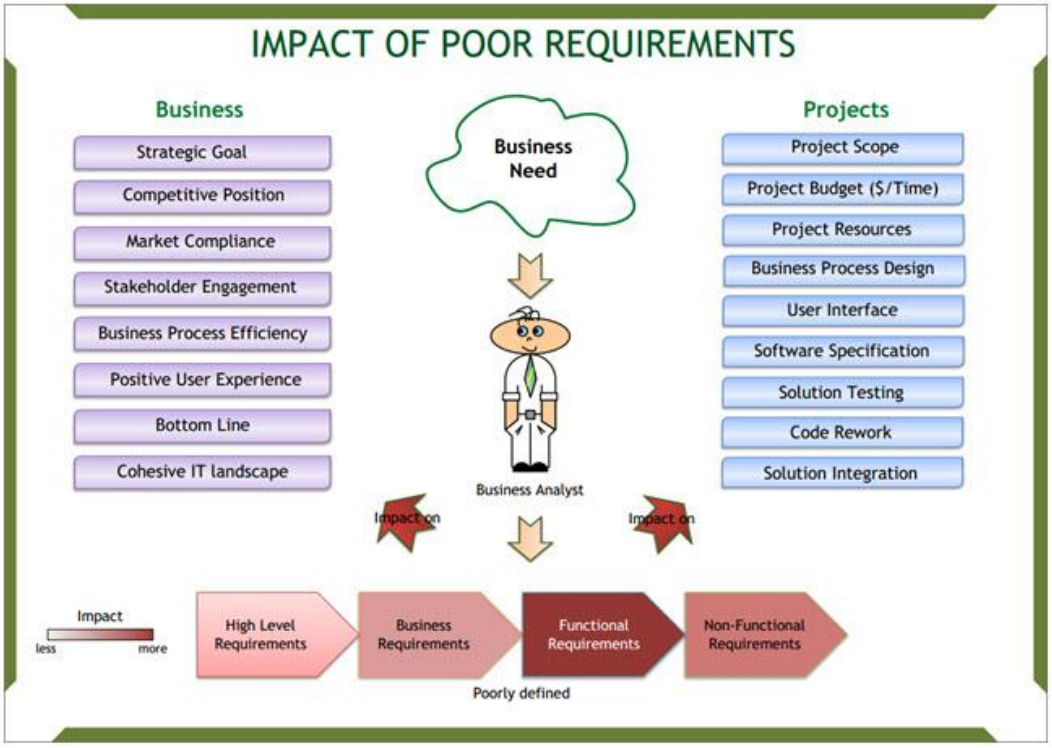


Figure 3 Impact of Poor requirement gathering process [1]

Projects are undertaken by the business to satisfy a strategic goal. Poor requirements have the following effects on projects (and subsequently impact the strategic goals of the business): [3]

- Scope creep negatively affecting budget and completion time
- Low utilzation of resources and higher overheads
- Inadequate business process design (due to insufficient details about activities)
- Poor design and ergonomics of the user interface, resulting in lower productivity
- Inadequate software specification, resulting in lower developer productivity
- Poor specification amplifies the negative effect of poor requirements when it comes to software testing, leading to higher costs and lower quality of the solution

- Time-consuming and costly code rework
- Difficulties in solution integration.

❖ Input phase of CNMRSF Model (From Chapter 5 Suggested Model)

Input phase is divided into three steps: Selection of Type of Requirement Document, Getting Requirement Document and Calling Processing Module by providing requirement document. Fig 3 shows the working flow of input phase. In this phase, Requirement document can get from your local computer drive. Main functionality of this phase is to get exact type and requirement document for further analysis.

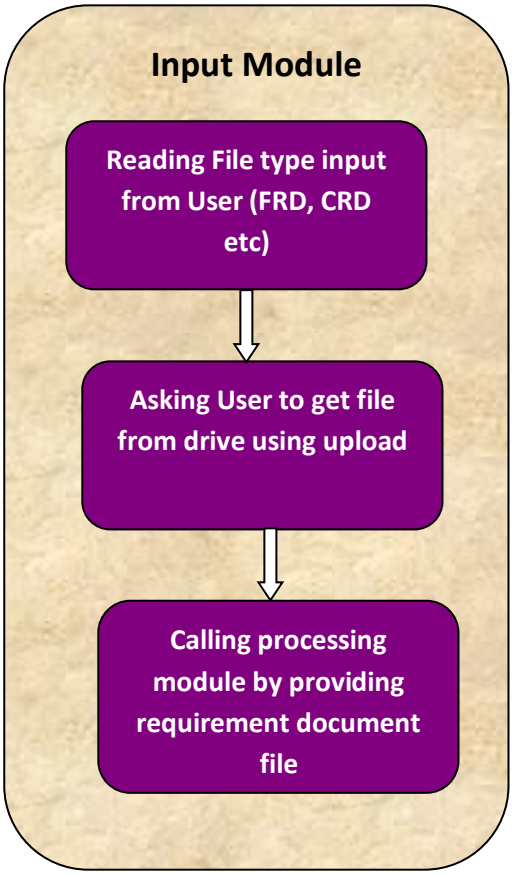


Figure 4 Input module of CNMRSF Model

Annexure -II

❖ Processing phase of CNMRSF Model

Processing phase deals with integrated functionality of Reading Requirement document, analyzing requirement document by call customer's Needs Engine (RA Engine) and Execution of Output module to generate list of Issues, Impacts and Actions for particular type of requirement issue.

In processing phase, RMEngine gets call to analyze exact requirement issue type. Functionality of RMEngine is explained in detail in below section. After executing RMEngine, processing module checks type of requirment issue provided by RMEngine, and based on that it calls output module. While calling output module, requirment type is getting provided as an input argument. Following algorithm should get executed by Processing module to call Output module.

```
If requirment type = Incomplete
{
Call Output Module(Incomplete);
}
If requirment type = Incorrect {
Call Output Module(Incorrect);
}
If requirment type = Ambiguity{
Call Output Module(Ambiguity);
}
If requirment type = Volatilty{
Call Output Module(Volatilty);
}
If requirment type = Tracability{
Call Output Module(Tracability);}
```

Annexure -II

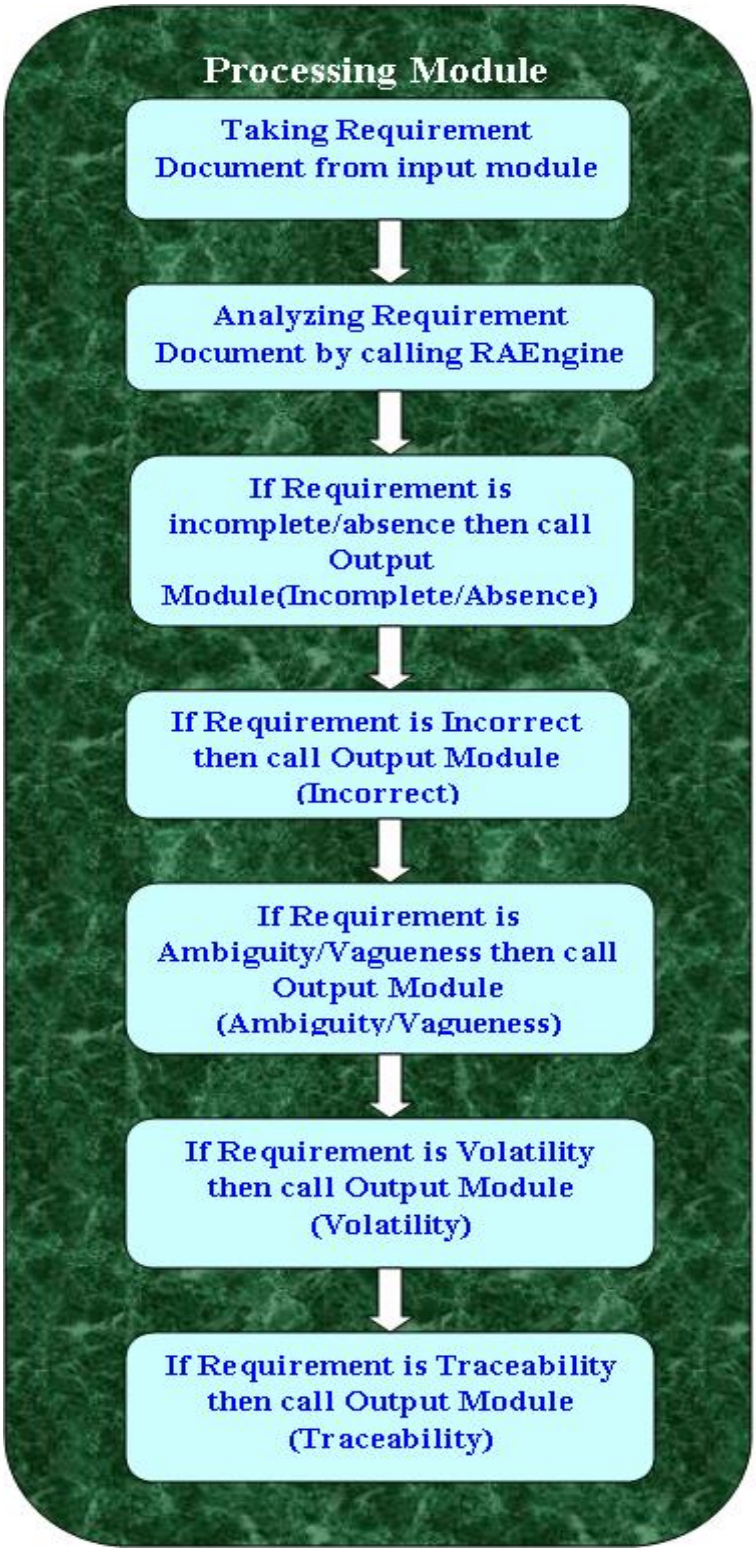


Figure 5. Processing module of CNMRSF Model

Annexure -II

Here based on input argument provided in call of Output Module, List of Issues, Impacts and Actions are getting displayed by Output module. Output Module has been explained in detail in below "Output phase of CNMRSF Model" section.

❖ **Requirement Analysis Engine (RAEngine)**

Requirement Analysis Engine is used in processing phase. It is developed based on the requirement issues responded by respondent from different software companies. For RMEngine, following requirement issues has been considered [4].

1. Incomplete/Absent
2. Incorrect
3. Ambiguity & Vagueness
4. Volatility
5. Traceability

RAEngine is heart of CNMRSF module. Without RAEngine, CNMRSF can not do anything. RAEngine basically works on if else ladder concept. It first checks what is exact requirement issue present in provided requirement document and based on that decide type of requirement issue. For deciding appropriate requirement issue, RAEngine analyze requirement document by compairing it with software system architechure document and tries to provide exact requirement issue. RAEngine takes Requirement document as an input and generates requirement issue type by considering many isssues present in provided requirement document.

As metioned above, RAEngine maninly focuses on five type of requirement issues like incomplete,incorrect, Ambuiguity,vaguess, volatility and traceability etc. [4]. Based on

different conditions like if track table is missing or improper change control process found then its mark requirement issue as traceability issue. If functional or non-functional requirements are missing then it marks requirement issue type as incorrect. If there is change between old requirement document and new requirement document then it marks requirement issue as volatility. If requirement followed poor requirement definition then it marks requirement issue as Ambiguity and Vagueness. Like wise it checks for Incomplete/Absence requirement issue. Here using if syntax RAEngine verifies many conditions to decides appropriate requirement issue.

Once requirement issue is identified by RAEngine, it returns that requirement issue back to processing module and then processing module works on further analysis.

Annexure -II

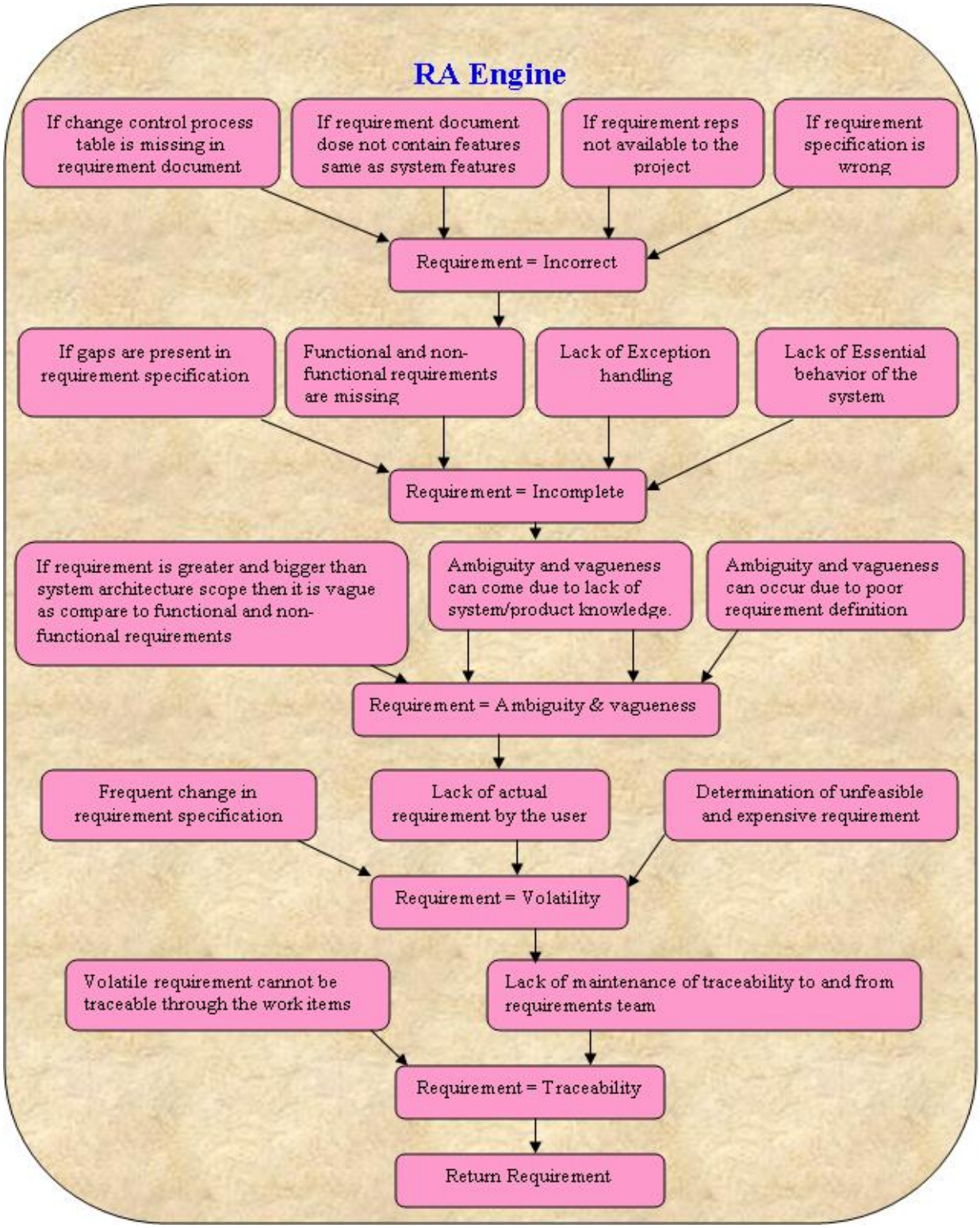


Figure 6. Customer's Needs Engine used in CNMRSF Model

Annexure -II

❖ **Output Phase of CNMRSF model**

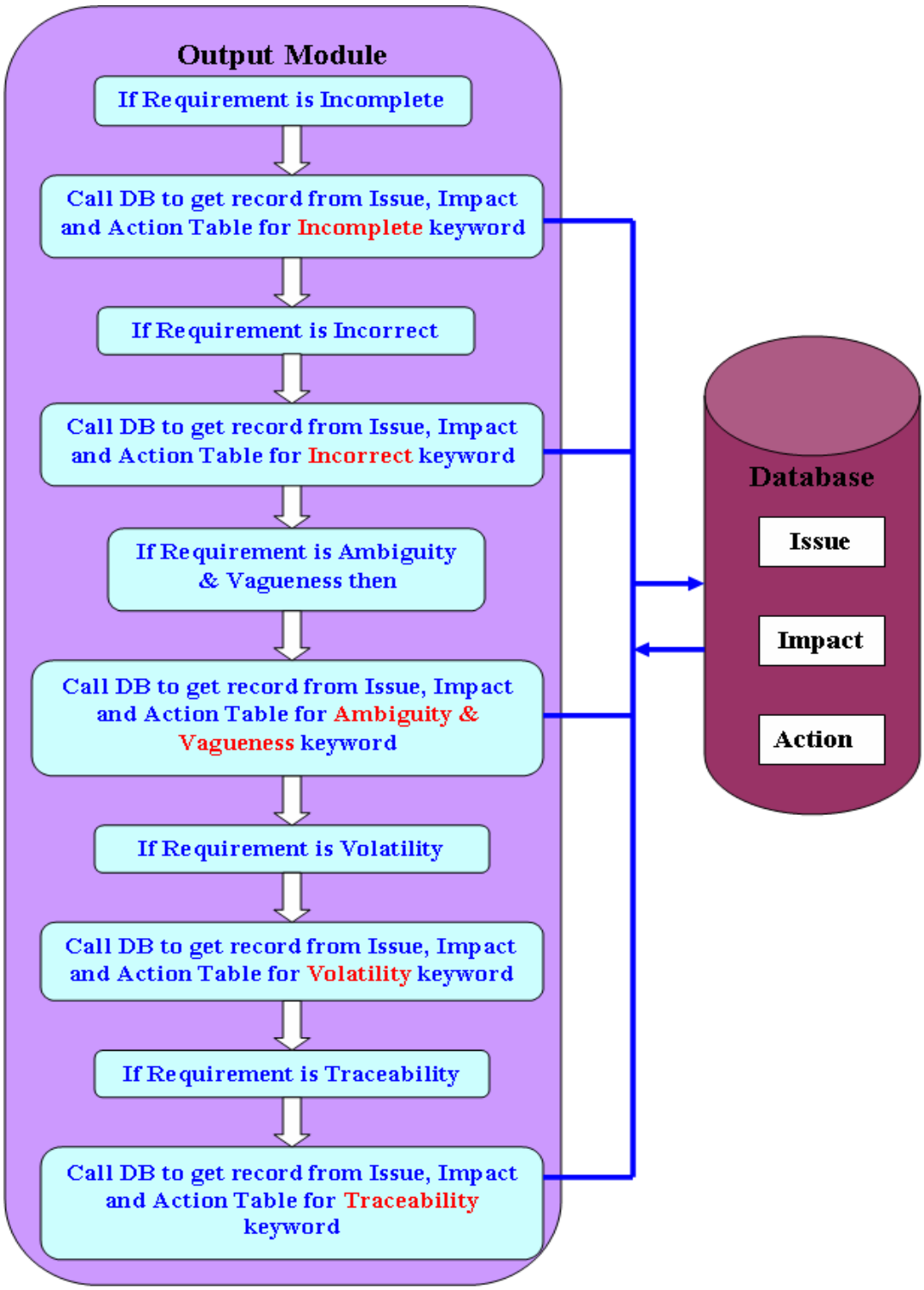


Figure 7. Output Module of CNMRSF Model

Annexure -II

Output phase deals with generation of output based on input argument as a requirement issue type provided by processing phase. Output phase has module named as Output Module and this module is basically gets executed by Processing module. Main functionality of output module is to get requirement type issue as an input and based on this input query to database to fetch corresponding list of Issues, Impacts and Action points. This module displaying list of Issues, Impacts and Actions based on corresponding requirement issue type.

Output module deals with database to fetch records from three different tables named as Issue, Impact and Action. These three tables have following schema.

Issue Table

Issue Name	Issue Description

Impact Table

Impact Name	Impact Description

Action Table

Action Name	Action Description

Annexure -II

SQL Queries to create these three tables:

Create Table Issue(Issue Name varchar(20), Issue Description varchar(100));

Create Table Impact(Impact Name varchar(20),Impact Description varchar(100));

Create Table Action(Action Name varchar(20), Action Description varchar(100));

These tables should be created in database before executing CNMRSF model. This is prerequisite and important step to execute CNMRSF model.

Once output module gets call from processing module with a requirement issue type then it executes following queries to fetch corresponding records for requirement issue type. Output module uses following SQL queries to fetch records from Issue, Impact and Action tables.

SQL queries for Issue Table:

Select * from Issue where Issue Name = “Incomplete”;

Select * from Issue where Issue Name = “Incorrect”;

Select * from Issue where Issue Name = “Ambiguity & Vagueness”;

Select * from Issue where Issue Name = “Volatility”;

Select * from Issue where Issue Name = “Traceability”;

Annexure -II

SQL queries for Impact Table:

Select * from Impact where Impact Name = “Incomplete”;

Select * from Impact where Impact Name = “Incorrect”;

Select * from Impact where Impact Name = “Ambiguity & Vagueness”;

Select * from Impact where Impact Name = “Volatility”;

Select * from Impact where Impact Name = “Traceability”;

SQL queries for Action Table:

Select * from Action where Action Name = “Incomplete”;

Select * from Action where Action Name = “Incorrect”;

Select * from Action where Action Name = “Ambiguity & Vagueness”;

Select * from Action where Action Name = “Volatility”;

Select * from Action where Action Name = “Traceability”;

For Issue, Impact and Action tables records, need to refer section 6.6.

Tables Definition

Issue, Impact and Action tables should have following list of records in database.

Annexure -II

- **Issue Table**

Issue Name	Issue Description
Incomplete	Gaps will be present in Requirement specification
Incomplete	Functional and non-functional requirements are missing
Incomplete	UI layout and sequencing failure can come
Incomplete	Essential behaviour of system may fail
Incomplete	System performance issue can come
Incomplete	Data conversion issues
Incomplete	Lack of Exception Handling
Incorrectness	Requirement specification can be wrong or incorrect
Incorrectness	Business Reps may not be good developers or SMEs
Incorrectness	Business Reps may not be available to the project as required
Incorrectness	Inadequate change control process within project
Ambiguity & Vagueness	If requirement is greater and bigger than system architecture scope then it is vague as compare to functional and non-functional requirements
Ambiguity & Vagueness	Ambiguity and vagueness can come due to lack of system/product knowledge. BA's not knowing exact details of the requirements

Annexure -II

Ambiguity & Vagueness	Ambiguity and vagueness can occur due to poor requirement definition process
Volatility	Frequent change in requirement specification
Volatility	Lack of actual requirement by the user
Volatility	Determination of unfeasible and expensive requirement
Traceability	Volatile requirement cannot be traceable through the work items of project.
Traceability	Lack of maintenance of traceability to and from requirements team

Table 1 Issue table used by CNMRSF Model

- **Impact Table**

Impact Name	Impact Description
Incomplete	Incomplete requirement can lead to gaps in software testing
Incomplete	More system failures can occur in customer system like production , UAT
Incomplete	System testing can get delayed
Incomplete	Estimations for software testing can be inaccurate
Incomplete	System performance issue can come
Incomplete	Testing team productivity and credibility can get decreased

Annexure -II

Incomplete	Delayed in Software benefit realization
Incorrectness	Incorrect test cases and test results will be created.
Incorrectness	Loss of time in investigation of test case failures
Incorrectness	Rework of test cases writing and re-testing.
Incorrectness	Loss of testing team efforts, time and credibility
Ambiguity & Vagueness	Loss of time in clarification of requirements
Ambiguity & Vagueness	Unable to determine how to test requirement
Ambiguity & Vagueness	Incorrect test cases and test results will be created.
Ambiguity & Vagueness	Rework of test cases writing and re-testing.
Ambiguity & Vagueness	Loss of testing team efforts, time and credibility
Volatility	Rework of test cases writing and re-testing
Volatility	Extended re-testing

Annexure -II

Volatility	Delay in delivery dates
Volatility	Delayed in Software benefit realization
Traceability	Unable to analyze impact of changing requirements
Traceability	Unknown test cases coverage
Traceability	Unable to evaluate number of requirements successfully met.
Traceability	Unable to evaluate which requirement have not been successfully delivered

Table 2 Impact table used by CNMRSF Model

- **Action Table**

Action Name	Action Description
Incomplete	Need to use supplement testing with experience based techniques
Incomplete	Relationship between testing and requirement team should be good,
Incomplete	Testing team must understand the business problems and solutions made for these problems
Incomplete	Early testing should be implemented
Incomplete	Checklist for Requirement specification reviews should be developed
Incorrectness	Raise risk regarding unavailability of Developers or SMEs

Annexure -II

Incorrectness	Testing team should involve in all requirement specific meetings, workshops and sessions
Incorrectness	Ask for central change management process so that testing team can review change.
Incorrectness	Maximum communication should be happened between testing, development and requirement teams.
Ambiguity & Vagueness	Need to have risk based testing.
Ambiguity & Vagueness	Need to start early testing and reviews test cases and test results bits early.
Ambiguity & Vagueness	Need to use critical thinking technique.
Ambiguity & Vagueness	Need to involve tester or test analyst or test lead (or complete testing team) in requirements gathering meetings, workshops and JAD sessions
Volatility	Scenarios based testing instead of requirements based testing.
Volatility	Need to follow good change management practices
Volatility	Need to have exploratory testing

Annexure -II

Volatility	Need to work on building consensus with the business representatives on the expected test result
Traceability	Need to implement and maintain a requirements traceability matrix
Traceability	Need to implement requirement change management process
Traceability	Need to assign unique identifiers to requirements
Traceability	Need to make test system component automate so that it will save testing time even if requirement gets changed.

Table 3 Action table used by CNMRSF Model

Requirement gathering is the process of collection of requirement from the client but many times due to incorrect requirement collection complete software project gets impacted. This chapters provides detail understanding about reasons behinds poor or incorrect requirements and how it impacts on software project and customer's business.

Annexure -II

References:

1. Chapter 4 'Capturing the Requirements', http://www.cse.msu.edu/~chengb/RE-491/Papers/atlee-chapter4.pdf
2. http://www.tutorialspoint.com/software_engineering/software_engineering_overview.htm
3. http://www.jot.fm/issues/issue_2007_01/column2.pdf
4. http://www.enterprisecioforum.com/en/blogs/pearl/it-project-failure-symptoms-and-reasons.
5. Indika Perera, "Impact of Poor Requirement Engineering in Software Outsourcing: A Study on Software Developers' Experience". Int. J. of Computers, Communications & Control, ISSN 1841-9836, E-ISSN 1841-9844 Vol. VI (2011), No. 2 (June), pp. 337-348
6. Requirements Engineering A good practice guide, Ramos Rowel and Kurts Alfeche, John Wiley and Sons, 1997
7. Donald Firesmith**,** Software Engineering Institute, U.S.A **"**Common Requirements Problems, Their Negative Consequences, and the Industry Best Practices to Help Solve Them". http://www.jot.fm/issues/issue_2007_01/column2/
8. Christel, Michael and Kyo C. Kang (September 1992). "Issues in Requirements Elicitation". Technical Report CMU/SEI-92-TR-012. CMU / SEI. Retrieved January 14, 2012.
9. https://www.utdallas.edu/~chung/RE/Getting_requirements_right-avoiding_the_top_10_traps.pdf

REFERENCES

1. The Standish Group Report (CHAOS). (2003). Retrieved November 2013, from :http://www.projectsmart.co.uk/docs/chaos-report.pdf.
2. Boehm, B., & Bose, P. (1994). A collaborative spiral software process model based on Theory W. Third International Conference on the Software Process, pp. (59-68).
3. Cao, L., & Ramesh, B. (2008). Agile Requirements Engineering Practices: An Empirical Study. IEEE Software, 25 (1), pp. (60-67).
4. 2013, Md Rounok Salehin "Missing Requirements Information and its Impact on Software Architectures:A Case Study" The School of Graduate and Postdoctoral Studies The University of Western Ontario,London, Ontario, Canada
5. Gross, A., Doerr, J. (2012). What do software architects expect from requirements specifications? results of initial explorative studies. IEEE First International Workshop on Twin Peaks of Requirements and Architecture, IEEE Software, pp. (41-45).
6. Lee, S., & Rine, D. (2004). Missing Requirements and Relationship Discovery through Proxy Viewpoints Model. 19th annual ACM Symposium on Applied Computing, pp. (1513-1518). Nicosia, Cyprus.
7. George, B., Bohner, S. A., & Prieto-Diaz, R. (2004). Software information leaks: A complexity perspective. Ninth IEEE International Conference on Engineering Complex Computer Systems (ICECCS'04), pp. (239-248). Florence, Italy: IEEE Computer Society.
8. Gumuskaya, H. (2005). Core Issues Affecting Software Architecture in Enterprise Projects. Proceedings of World Academy of Science, Engineering And Technology, volume 9, pp. (35-41)
9. Thomas Kiihne , "What is Model?" Darmstadt University of Technology, Darmstadt, Germany.
10. http://www.la1.psu.edu/cas/jgastil/pdfs/conceptualdefinitiondeliberation.pdf

11. Kirsten Kiefer, “The Impact of Requirement issues on testing”, Software Education associates Ltd
12. Brooks, F. 1987. No Silver Bullet: Essence and Accidents of Software Engineering. IEEE Computer, Vol. 20, No. 4, April 1987, 10-19.
13. Jayaswal, B. K., Patton, P. C. 2006. Design for Trustworthy Software: Tools, Techniques, and Methodology of Developing Robust Software, 1st Edition. (September 2006), Prentice Hall edition.
14. Zowghi, D. 2002. A Study on the Impact of Requirements Volatility on Software Project Performance. Proceedings of Ninth Asia-Pacific SE Conference (APSEC‟ 2002), IEEE Computer Science.
15. Taghi, M., Khoshgoftaar, N., Sundaresh, N. 2006. An empirical study of predicting software faults with case-based reasoning. (June 2006), Software Quality Control.
16. Jiang, Y., Cukic, B., Ma, Y. 2008. Techniques for evaluating fault prediction models. (October 2008), Empirical Software Engineering.
17. M.P.Singh, Rajnish Vyas, “Requirements Volatility in Software Development Process” International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-2, Issue-4, September 2012
18. Zowghi, N. Nurmuliani, ―A study of the Impact of requirements volatility on Software Project Performance, Proceedings of the Ninth Asia-Pacific Software Engineering Conference , APSEC 2002, Gold Cost, Queensland, Australia,04-06 Dec 2002, pp:3-11.
19. http://www.mapsofindia.com/maps/maharashtra/pune.htm (23/7/2008)
20. http://www.mapsofindia.com/pune/software-company-pune.html
21. https://maps.google.co.in/maps?hl=en-IN&gbv=2&ie=UTF-8&fb=1&gl=in&q=software+companies+in+pune&hq=software+companies&hnear=0x3

bc2bf2e67461101:0x828d43bf9d9ee343,Pune,+Maharashtra&ei=av_nVKH6O86IuwSvrIL4Bw&ved=0CB4QtQM&output=classic&dg=brw

22. Sr. S. P. Gupta, “Statistical Methods”, Sultanchand & Sons Publication, New Delhi.
23. Don Gotterbarn, “Reducing Software Failures: Addressing the Ethical Risks of the Software Development Lifecycle” Australian Journal of Information Systems
24. Muhammad Naeem Ahmed Khan and et.all (2013), “Review of Requirements Management Issues in Software Development” I.J.Modern Education and Computer Science, 2013, 1, 21-27, Published Online January 2013 in MECS (http://www.mecs-press.org/) DOI: 10.5815/ijmecs.2013.01.03
25. https://web.cs.dal.ca/~hawkey/3130/SEBackground4.pdf
26. Systems Development Lifecycle: Objectives and Requirements. Bender RPT Inc. 2003
27. Vanshika Rastogi (2015), “Software Development Life Cycle Models- Comparison, Consequences” IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 6 (1) , 2015, 168-172
28. Software Development Life Cycle (SDLC) Yogi Berra presentation
29. Ms. Shikha maheshwari and Prof.Dinesh Ch. Jain 2012 “A Comparative Analysis of Different types of Models in Software Development Life Cycle” International Journal of Advanced Research in Computer Science and Software Engineering, Volume 2, Issue 5, May 2012
30. Royce, Winston (1970), "Managing the Development of Large Software Systems" (PDF), Proceedings of IEEE WESCON 26 (August): 1–9
31. PK.Ragunath, S.Velmourougan, P. Davachelvan, ,S.Kayalvizhi, R.Ravimohan (2010) “Evolving A New Model (SDLC Model-2010) For Software Development Life Cycle (SDLC)” IJCSNS International Journal of Computer Science and Network Security, VOL.10 No.1, January 2010

32. Seema, Sona Malhotra 2012 “Analysis and tabular comparision of popular SDLC models” International Journal of Advances in computing and Information Technology.
33. Sonali MAthur and Shaily Malik (2010), “Advancements in the V-Models”, International Journal of Computer Applications (0975-8887) Volume 1- No.12
34. Naresh Kumar, A. S. Zadgaonkar, Abhinav Shukla “Evolving a New Software Development Life Cycle Model SDLC-2013 with Client Satisfaction” International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-3, Issue-1, March 2013
35. G. Kotonya and I. Sommerville, (1998), have published article on “Requirements Engineering: Processes and Techniques”, in the book published by Chichester, UK: John Wiley & Sons.
36. Requirements Engineering A good practice guide, Ramos Rowel and Kurts Alfeche, John Wiley and Sons, 1997
37. I. Sommerville and P. Sawyer (1997), Requirements Engineering: A Good Practice Guide, New York: John Wiley & Sons,.
38. http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=948567&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D948567
39. K. E. Wiegers, Software Requirements, 2nd ed., Redmond, W A: Microsoft Press, 2003.
40. http://prr.hec.gov.pk/Chapters/369S-2.pdf (Software Testing reference)
41. Donald Firesmith, Software Engineering Institute, U.S.A “Common Requirements Problems, Their Negative Consequences, and the Industry Best Practices to Help Solve Them”. http://www.jot.fm/issues/issue_2007_01/column2/
42. Olga Liskin , et al., “Supporting Acceptance Testing in Distributed Software Projects with Integrated Feedback Systems: Experiences and Requirements” 2012 IEEE Seventh International Conference on Global Software Engineering

43. Vishawjyoti, Sachin Sharma, “Study and Analysis of automation testing techniques” , Journal of global research in computer science, Volume 3, No. 12, December 2012, ISSN-2229-371
44. Antonia Bertolino has published his research article on “Software testing research and practice”
45. Vivek Kumar (2012) has published his article on “Comparison of Manual and automation testing” International Journal of Research in Science And Technology, (IJRST) 2012, Vol. No. 1, Issue No. V, Apr-Jun, ISSN: 2249-0604
46. R. M. Sharma (2014), “Quantitative Analysis of Automation and Manual Testing” International Journal of Engineering and Innovative Technology (IJEIT) Volume 4, Issue 1, July 2014
47. Harsh Bhasin, at.el. (2014), have published research article on “Black Box Testing based on Requirement Analysis and Design Specifications”
48. MIRZA MAHMOOD BAIG (2009), “NEW SOFTWARE TESTING STRATEGY” N.E.D. University of Engineering & Technology
49. Mohd. Ehmer Khan , Farmeena Khan “A Comparative Study of White Box, Black Box and Grey Box Testing Techniques”, (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 3, No.6, 2012
50. Paul C. Jorgensen (2013) published book on “Software testing: a craftsman's approach” CRC Press
51. Wasif Afzal et al. (2008) “A Systematic Mapping Study on Non-Functional Search-based Software Testing” paper available at http://www.researchgate.net/publication/221391274
52. W. K. Chan et al (2002), have published article on “An Overview of Integration Testing Techniques for Object-Oriented Programs” Proceedings of the 2nd ACIS Annual

International Conference on Computer and Information Science (ICIS 2002), International Association for Computer and Information Science, Mt. Pleasant, Michigan (2002)

53. Shivkumar Hasmukhrai Trivedi, (2012), has published research article on "Software Testing Techniques" International Journal of Advanced Research in Computer Science and Software Engineering, Volume 2, Issue 10, October 2012

54. Leung, H.K.N (1989) has published article on "Insights into regression testing" Software Maintenance, 1989., Proceedings., Conference on

55. M. Fagan, "Design and Code Inspections to Reduce Errors in Program Development," IBM Systems Journal, vol. 38, no. 2/3, pp. 258–287, 1999.

56. Hitesh Tahbildar at el(2011). "Automated software test data generation: Direction of research" International Journal of Computer Science & Engineering Survey (IJCSES) Vol.2, No.1, Feb 2011

57. http://www.rishabhsoft.com/blog/beta-testing-the-importance (2011)

58. Ms. S. Sharmila, "Analysis of Performance Testing on Web Applications" International Journal of Advanced Research in Computer and Communication Engineering Vol. 3, Issue 3, March 2014

59. Pooja Ahlawat (2013) "A Comparative Analysis of Load Testing Tools Using Optimal Response Rate" International Journal of Advanced Research in Computer Science and Software Engineering. Volume 3, Issue 5, May 2013

60. Chapter 4 'Capturing the Requirements', http://www.cse.msu.edu/~chengb/RE-491/Papers/atlee-chapter4.pdf

61. 2013, Md Rounok Salehin “Missing Requirements Information and its Impact on Software Architectures:A Case Study” The School of Graduate and Postdoctoral Studies The University of Western Ontario,London, Ontario, Canada
62. Mohd. Ehmer Khan, “Different Forms of Software Testing Techniques for Finding Errors,” IJCSI, Vol. 7, Issue 3, No 1, pp 11-16, May 2010
63. Christel, Michael and Kyo C. Kang (September 1992). "Issues in Requirements Elicitation". Technical Report CMU/SEI-92-TR-012. CMU / SEI. Retrieved January 14, 2012.
64. Donald Firesmith, Software Engineering Institute, U.S.A “Common Requirements Problems, Their Negative Consequences, and the Industry Best Practices to Help Solve Them”. http://www.jot.fm/issues/issue_2007_01/column2/
65. Indika Perera, “Impact of Poor Requirement Engineering in Software Outsourcing: A Study on Software Developers’ Experience”. Int. J. of Computers, Communications & Control, ISSN 1841-9836, E-ISSN 1841-9844 Vol. VI (2011), No. 2 (June), pp. 337-348
66. Requirements Engineering A good practice guide, Ramos Rowel and Kurts Alfeche, John Wiley and Sons, 1997
67. Chapter 4 ‘Capturing the Requirements’, http://www.cse.msu.edu/~chengb/RE-491/Papers/atlee-chapter4.pdf
68. http://www.tutorialspoint.com/software_engineering/software_engineering_overview.htm
69. http://www.jot.fm/issues/issue_2007_01/column2.pdf
70. http://www.enterprisecioforum.com/en/blogs/pearl/it-project-failure-symptoms-and-reasons.

71. IndikaPerera, “Impact of Poor Requirement Engineering in Software Outsourcing: A Study on Software Developers’ Experience”. Int. J. of Computers, Communications & Control, ISSN 1841-9836, E-ISSN 1841-9844 Vol. VI (2011), No. 2 (June), pp. 337-348
72. Requirements Engineering A good practice guide, Ramos Rowel and KurtsAlfeche, John Wiley and Sons, 1997
73. **Donald Firesmith,** Software Engineering Institute, U.S.A **“**Common Requirements Problems, Their Negative Consequences, and the Industry Best Practices to Help Solve Them”. http://www.jot.fm/issues/issue_2007_01/column2/
74. Christel, Michael and Kyo C. Kang (September 1992). "Issues in Requirements Elicitation". Technical Report CMU/SEI-92-TR-012. CMU / SEI. Retrieved January 14, 2012.
75. https://www.utdallas.edu/~chung/RE/Getting_requirements_right-avoiding_the_top_10_traps.pdf
76. Vidya Gaveakr, (2013) “A Study of Geographic Information System based computerized framework to enhance the water supply system in Pune City” Thesis of Computer Management Dept., Tilak Maharashtra Univerisity.
77. http://www.enterprisecioforum.com/en/blogs/pearl/it-project-failure-symptoms-and-reasons.
78. K.K. Aggarwal and Yogesh Singh,”Software Engineering”,New age International Publishers, third Edition, 2008.
79. http://www.umsl.edu/~sauterv/analysis/Fall2010Papers/Isserman/
80. http://www.practicalecommerce.com
81. http://www.reqharbor.com/
82. https://www.google.co.in/webhp?hl=en#hl=en&q=data%20dictionary

83. https://ankitmathur111.wordpress.com/2012/06/20/whats-whys-hows-of-software-testing-wwh/
84. http://istqbexamcertification.com/what-is-fundamental-test-process-in-software-testing/
85. Thomas Kiihne , "What is Model?" Darmstadt University of Technology, Darmstadt, Germany.
86. http://www.la1.psu.edu/cas/jgastil/pdfs/conceptualdefinitiondeliberation.pdf
87. http://www.voltreach.com/Development_Methodologies.aspx
88. Kirsten Kiefer, "The Impact of Requirement issues on testing", Software Education associates Ltd.
89. http://www.softed.com/resources/docs/impact-of-requirements-issues-on-testing.pdf
90. http://blog.sei.cmu.edu/post.cfm/common-testing-problems-pitfalls-to-prevent-and-mitigate
91. http://kinzz.com/resources/articles/91-project-failures-rise-study-shows
92. S. Arun Kumar and T.Arun Kuma "Study The Impact Of Requirements Management Characteristics In Global Software Development Projects: An Ontology Based Approach" in International Journal of Software Engineering & Applications (IJSEA), Vol.2, No.4, October 2011
93. http://www.pune.ws/in/?list=it_software_companies-hinjawadi|here
94. http://www.pune.ws/in/?list=hadapsar-it_software_companies
95. http://www.pune.ws/in/?list=it_software_companies-kharadi
96. http://www.pune.ws/in/?list=it_software_companies-magarpatta
97. http://hiapune.in
98. https://www.dnb.co.in/TopIT/company_listing.asp?PageNo=1&q=employee&r
99. Software Engineering for Students- A Programming Approach by Douglas Bell Pearson Education. Pg 230-255

100. Ralph R. Young, Effective Requirements Practices. Addison-Wesley, 2001, page 108.

101. Kotonya, G. and Sommerville, I. 1998. Requirements Engineering: Processes and Techniques Chichester, UK: John Wiley and Sons.

102. Karl E. Wiegers More About Software Requirements: Thorny Issues and Practical Advice(Microsoft Press, 2006; ISBN 0-7356-2267-1)Chapter 2: Truths About Software Requirement

103. Henry Johnson, An approach to software project management through requirements engineering, At Texas Tech University, Henry Johnson, December 2010

104. Davis , C.J, Fuller, R.M. Tremblay, M.C. & Berndt, D.J. (2006). Communication Challenges in requirements elicitation and the use of the repertory grid technique. Journal of computer information Systems, 78

105. Bourque, P.; Fairley, R.E. (2014). "Guide to the Software Engineering Body of Knowledge (SWEBOK)". IEEE Computer Society. Retrieved 17 July 2014

106. Software Engineering – A Practitioners Approach by Roger S. Pressman Tata McGraw Hill.

Quality, 5th ed., Prentice-Hall, 2010. Donna C. S. Summers. Pg 20-57

107. Total Quality Management, Prentice Hall, 2003 Dale H. Besterfield. . Pg 37-77

108. Information Technology Project Management -Kathy Schwalbe. Pg 7-177

109. Software Metrics A rigorous and practical approach – N Fenton, S Lawrence Pfleeger.

Pg 170-255

110. Research Methodology Methods and Techniques By C R Kothari and Gaurav Garg.

Pg 52-109

111. A Practitioner's Guide to Software Test Design, Lee Copeland, 2003. Pg 38-97

112. The Art of Software Testing, 2nd edition, Glenford Myers, et. 2004. Pg 76- 143

113. Software Testing Techniques, 2nd edition, Boris Beizer, 1990. Pg 14-65

114. How to Break Software: A Practical Guide to Testing, James Whittaker, 2002. Pg123-254

115. http://technosoftwares.com/software-development-life-cycle

116. http://www.jamasoftware.com/blog/change-impact-analysis-2

117. "Impact of software requirement volatility pattern on project dynamics: evidences from a case study" International Journal of Software Engineering & Applications (IJSEA), Vol.2, No.3, July 2011

Books

1. Software Engineering for Students- A Programming Approach by Douglas Bell Pearson Education. Pg 230-255
2. Software Engineering – A Practitioners Approach by Roger S. Pressman Tata McGraw Hill.
3. Quality, 5th ed., Prentice-Hall, 2010. Donna C. S. Summers. Pg 20-57
4. Total Quality Management, Prentice Hall, 2003 Dale H. Besterfield. . Pg 37-77
5. Information Technology Project Management -Kathy Schwalbe. Pg 7-177
6. Software Metrics A rigorous and practical approach – N Fenton, S Lawrence Pfleeger. Pg 170-255
7. .Research Methodology Methods and Techniques By C R Kothari and Gaurav Garg. Pg 52-109
8. .A Practitioner's Guide to Software Test Design, Lee Copeland, 2003. Pg 38-97

9. The Art of Software Testing, 2nd edition, Glenford Myers, et. 2004. Pg 76- 143
10. Software Testing Techniques, 2nd edition, Boris Beizer, 1990. Pg 14-65
11. How to Break Software: A Practical Guide to Testing, James Whittaker, 2002. Pg123-254