EMPIRICAL ANALYSIS OF QUALITY ASSURANCE IN CMMI LEVEL 2 FOR COMPLIANCE

by

Qurat-ul-Ain

2009-NUST-MSPHD- CSE(E)-08

MS-09 (SE)

Submitted to the Department of Computer Engineering in fulfillment of the requirements for the degree of

MASTER OF SCIENCE
in
SOFTWARE ENGINEERING

Thesis Supervisor

Dr. Shoab A. Khan

College of Electrical & Mechanical Engineering

National University of Sciences & Technology

2012

DECLARATION

I hereby declare that I have written this thesis wholly on the basis of my own efforts under the supervision of my supervisor Dr. Shoab A. Khan. All the sources used have been cited and by no mean the contents of this thesis have been plagiarized. No fraction of the work presented has been submitted in support of any application for any other degree of qualification to this or any other university or institute of learning.

____

Qurat-ul-Ain

ACKNOWLEDGMENTS

First and foremost, I am thankful to Allah Almighty for His blessings. It would have been impossible to achieve this milestone without His consent.

I am heartily thankful to my Supervisor, Dr. Shoab Ahmed Khan, whose encouragement, supervision, and unwavering dedication, right from the preliminary to the concluding level, facilitated me to accomplish this work.

I am highly obliged to Mr. Farrukh Hashmi, who burnt midnight oil with me, and was a true support in all hard times. Have my gratitude. My heartiest thanks to Mr. Asad Waqar Malik, for the constant reminders, help and inspiration. Also I owe Thank you to the faculty and staff members of Department of Computer Engineering, EME, NUST, for their assistance.

I am eternally grateful to the Director at my work place, Mrs. Shaista Shahid, for understanding and flexibility she showed during the office hours and ultimately motivating me to achieve my goal.

I am indebted to my Father, M.Hanif and my Mother, Nasira Perveen for having faith in me and encouraging me all the way through, my twin brothers M. Arsalan Naz and M. Melhan Naz, who always stand by my side in all my endeavors, all elders of my Family who constantly send their prayers my way and my dear friends Usman Ghani, Muhammad Waqas, Bushra Shareef, Nazish Asad, Saddaf Rubab, Zeeshan Shafqat, Nazish Shafqat, Sehrish Shafqat, Sharjeel Butt, Junaid Anwar, Saima Ch, Hassan Ilyas, and Sumera Ch, who supported me either morally or practically during this long and eventful journey.

Last but not the least, a very special thank you to my loving fiancé Hassan for the care and support, during every odyssey of my life.

To my loving Parents, Brothers and Fiancé…

Who always keep me grounded…

ABSTRACT

The cost of software errors or problems as a result of quality lapse is a major difficulty to the software industry at large, to the developers of the Software as well its clients. The task of improving quality is therefore vital.

This thesis stands as a guideline for defining a model supporting Quality assurance program for CMMI level 2 certified company projects, by doing the empirical analysis of a CMMI Level 2 software project. In this maneuver, proposing a model for the assurance of quality, facilitating organizational strategy toward process and project improvement and achieving the quality objectives is the main focus. The whole model, after a complete fulfillment, would provide the organization with a guideline to achieve the organizational quality assurance objectives. It is vital to mention that this template itself does not improve the processes. It only shows the status of the chosen project/process after having the filled template executed. What this template generates is to provide the stakeholders with necessary information and basis to make informed decision afterwards in order to improve the chosen processes/projects. Since the processes in "Maturity Level 2 of CMMI" are project based, it is important that the status of processes quality be assured. This procedure plays a crucial role in creating a platform for moving to the next maturity level.

This research is dual process; firstly an empirical study of the projects is done and then based on the analysis of that empirical study a model is defined to make quality assurance better in an organization. Based on the approach of Empirical study and proposed model a Tool, one of its kind, Quality Assurance Gap Analyzer (QAGA) is developed, which takes Requirements, Test cases and their status as Input and generates a Report telling the user about the Gaps that exist in each requirement and an overall Gap Analysis of the whole project. It also generates a graph plotting Requirements on the basis of their criticality against the percentages of Gap that exists in them.

Results are helpful for the organizations to know the status of their Software Processes, Testing coverage of Requirements and thus a better understanding of Quality Assurance of the product.

LIST OF TABLES

Table of Contents

List of Figures

LIST OF ABBREVIATIONS

QA	Quality Assurance
SQA	Software Quality Assurance
PPQA	Process and Product Quality Assurance
CMMI	Capability Maturity Model Integration
SEI	Software Engineering Institute
TSP	Team Software Process
QAGA	Quality Assurance Gap Analyzer

Chapter 1

INTRODUCTION

For the last many years immense effort has been spent on gathering knowledge about software processes. There are Software Quality standards (such as CMMI, IEEE 12207, and ISO/IEC 90003 etc.) and some more verified literature. All these were extremely helpful in aiding developers deal with the complex problems of software processes, and ultimately the production of efficient and reliable software products. Quality improvements have a great impact on the performance of any operation in many aspects including improving productivity, growing revenue and minimizing costs.

There is limited research about CMMI quality assurance process area. One such effort has been put forward, presenting a tool, based on the design of a generic model evolved from the empirical study of a CMMI Level 2 Software Project, aiding in the assurance of the compliance of the CMMI Level 2 software projects based on of Gap analysis.

1.1. PROBLEM OVERVIEW

Quality Assurance is a heavily studied domain and researchers have been successful in proposing models and frameworks to address the issues that exist and achieve quality in a better and efficient way. But when it comes to CMMI Level 2 Process and Product Quality Assurance

Process area, no model has been yet proposed which may help the organizations in achieving CMMI level 2 in a faster and efficient manner.

Therefore, a model needed to be designed that can help organization to satisfy the goals of Process and product Quality Assurance Process Area and hence move towards the next level of CMMI.

1.2. PROJECT OBJECTIVES

The focus of this thesis is on the empirical study and thus improvement of the testing & quality assurance task in CMMI level 2 companies with software quality problems. Testing practices and quality assurance methods will be outlined in the thesis, explaining what to be used during the software quality improvement process in the company. Following the quality improvement process in the company a framework for improving software quality is produced.

1.3. THESIS OUTLINE

This thesis is a multi step process; firstly an empirical study of the CMMI Level 2 Software Project is done and after analyzing the empirical study, a generic model is defined, then a tool (QAGA) is developed and finally the results generated by QAGA are presented.

Adopting a qualitative research approach, firstly thorough study of literature regarding Software Process Improvement and CMMI is done. Then, a CMMI level 2 software project is taken from a Software Development Organization and that software project is empirically analyzed. With the help of this knowledge a generic model is evolved defining its components at attribute level. Third, a tool has been coded to examine the effectiveness of Model.

A case study approach is adopted, testing of projects is done with the help of that tool and the gaps in the Quality of Software is identified resulting in better understanding of the quality assurance of the software project.

In Chapter 2, description about quality, quality assurance and CMMI has been presented. Chapter 3 encompasses the related work that has been carried out in the same field. Chapter 4 covers the methodology of research, and finally in Chapter 5 Implementation and results are presented.

Chapter 2

BACKGROUND

2.1. INTRODUCTION

This chapter is detailed description of the concepts related to Quality and CMMI. The chapter begins with the presentation of standard definitions of Quality, Software Quality and Software Quality Assurance, followed by the description of Software Quality Assurance Standards. Then detail about CMMI and its Levels is discussed. Then, its key Process areas are presented. Then, a brief text on why CMMI is best reputed among all the Quality standards. Then description about the process area under consideration, which is "Process and Product Quality Assurance", is elaborated. In the end Generic Goals and Generic practices of CMMI Level 2 are discussed.

2.2. QUALITY

The term "Quality" is defined as: The entirety of functions and characteristics of a product that has to fulfill the requirements which are given. [1]

2.3. SOFTWARE QUALITY

In the field of Software Engineering, the definition of Software Quality is based on the two closely related philosophies.

a) Software Functional quality tells us how well it has fulfilled the functional requirements specified. Also called Fitness for Purpose.

b) How well it fulfills the non functional requirements. Also determine to what extent of correctness, software has been developed [2].

Quality of software can also be defined as the degree to which it conforms to the given criteria. Quality Criteria includes, but is definitely not limited to [3]:

i. Economy

ii. Correctness

iii. Resilience

iv. Integrity

v. Reliability

vi. Usability

vii. Documentation

viii. Modifiability

ix. Clarity

x. Understandability

xi. Validity

xii. Maintainability

xiii. Flexibility

xiv. Generality

xv. Portability

xvi. Interoperability

xvii. Testability

xviii. Efficiency

xix. Modularity

xx. Reusability

2.4. QUALITY ASSURANCE

To fulfill a product's quality requirements, activities are first planned and then implemented in a Quality System. These systematic activities are referred as Quality Assurance [4].

It is believed that Quality Assurance is different from Quality Control. In Quality Control, the main focus is the outputs of the processes. On the other hand, Quality Assurance is the name of comparison with the standards, orderly measurement and a feedback loop that takes care of error prevention [5].

Two main objectives of the Quality Assurance are:

i. Product should fulfill the intended purposes.

ii. Mistakes which are encountered or highlighted during testing should be removed.

Quality is determined by the users of the products and it has got nothing to do with the Cost and the descriptions of the products [6].

2.5. SOFTWARE QUALITY ASSURANCE

Software Quality Assurance is defined as the activities carried out to scrutinize the methods that are used in determining the assurance of Quality. There are several Models and Methods which are developed for the cause [7].

SQA assures that some check has been performed to guarantee that the product will work as it is required but it is applied to the code or non code artifacts. Software Quality Assurance covers the whole Software Development Process [9]

The organization of SQA is done as follows:

1.	Goals
2.	Commitments
3.	Abilities
4.	Activities
5.	Measurements
6.	Verifications

Table 1 Organization of SQA [8]

2.6. SOFTWARE QUALITY STANDARDS

Many Software Quality Assurance (SQA) standards have been written for various industries. Software organizations have to take care of the expectations of all type of customer. All these standards focus on developing an SQA standard practices that developers may adopt either willingly or keeping in view the demands of the customers. [10].

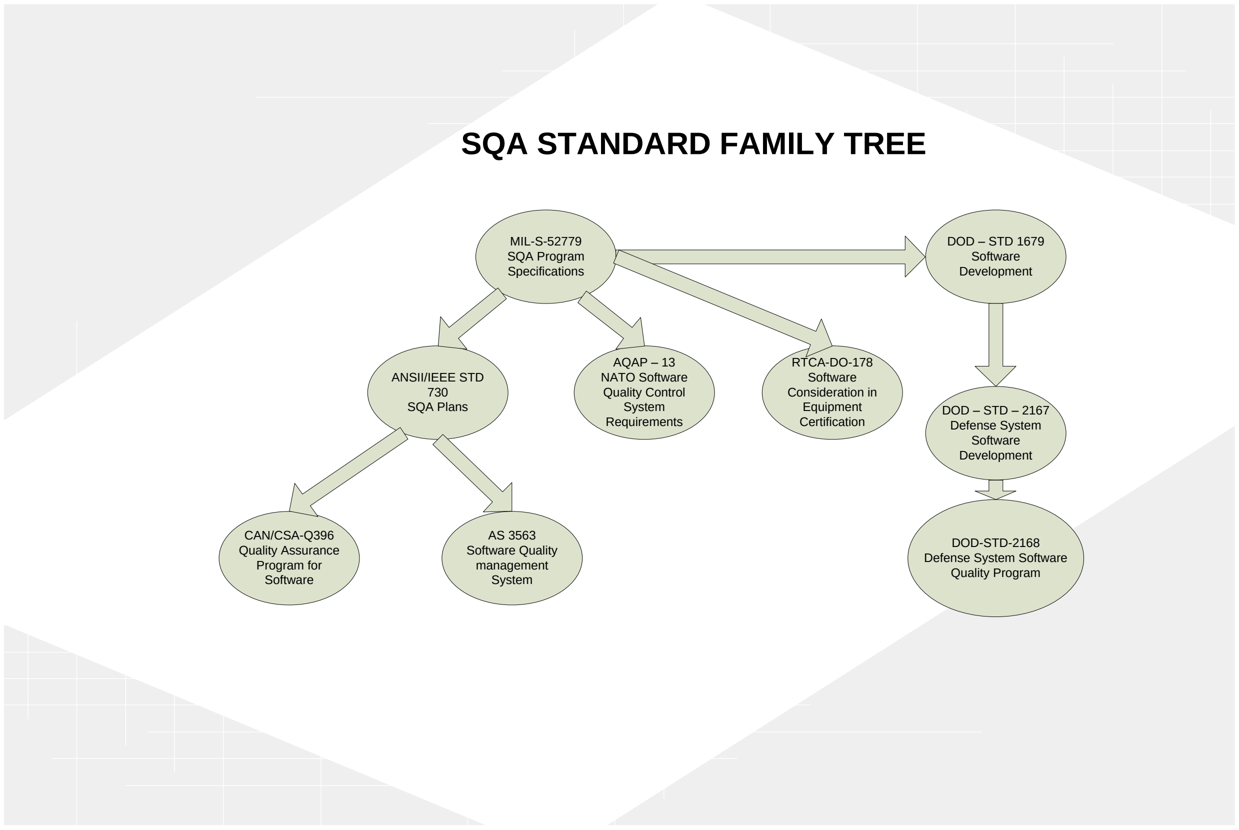

SQA STANDARD FAMILY TREE
MIL-S-52779 SQA Program Specifications
DOD – STD 1679 Software Development
ANSII/IEEE STD 730 SQA Plans
AQAP – 13 NATO Software Quality Control System Requirements
RTCA-DO-178 Software Consideration in Equipment Certification
DOD – STD – 2167 Defense System Software Development
CAN/CSA-Q396 Quality Assurance Program for Software
AS 3563 Software Quality management System
DOD-STD-2168 Defense System Software Quality Program

Figure 1 SQA Family Tree [10]

The first SQA standard extensively used was MIL-S-52779 [11], prepared in the mid- 1970s and has been updated only one time after that (i.e. MIL-S-52779A). Definite advances and improvements can be found in subsequent standards; in many cases, understandability and organization are greatly improved.

2.7. CMMI

Capability Maturity Model is an improvement model that aids organizations in improving their processes. It tells the organizations what features are most effective in making their performance better. It also identifies the strengths and weaknesses of the processes being carried out in an organization. Moreover it helps in turning the weaknesses of an organizations process into its strengths [12].

CMMI was a maneuver of a group of professionals from different areas of industry and the SEI at Carnegie Mellon University [23]. CMMI is a U.S. Patent registered model and copyrighted under Carnegie Mellon University. According to the standard definition of CMMI given by Software Engineering Institute (SEI), "Combining already separated organizational functions, Which helps in making progress in the priorities and goals, provisioning of guidelines for quality assurance processes, and also providing standards for the process going on" [24].

CMM is a predecessor of CMMI. Inception of CMM took place in 1987 which kept on evolving till 1997. The first version of CMMI which is “CMMI version 1.1” was developed in 2001, following the release of second version which is “CMMI Version 1.2” in 2006, and then the third version of CMMI which is “CMMI Version 1.3” was developed in 2010. The main modifications in CMMI V1.3 [25] are:

i. Agile Software Development [26]

ii. expansions to high maturity practices [27]

iii. Staged and Continuous presentations [28]

The areas addressed by CMMI are:

i. CMMI for Development (CMMI-DEV)

ii. CMMI for Services (CMMI-SVC)

iii. CMMI for Acquisition (CMMI-ACQ)

Name	Specification
CMMI for Development (CMMI-DEV)	Product and service development
CMMI for Services (CMMI-SVC)	Service establishment, management, and delivery
CMMI for Acquisition (CMMI-ACQ).	Product and service acquisition

Table 2 The Interest Areas of CMMI

2.7.1. Presentations of CMMI

CMMI exists in two representations:

i. Continuous

ii. Staged [23]

In Continuous representation, user concentrates only on those exact processes which are vital for organization's immediate business objectives and those with which a high Risk factor is associated.

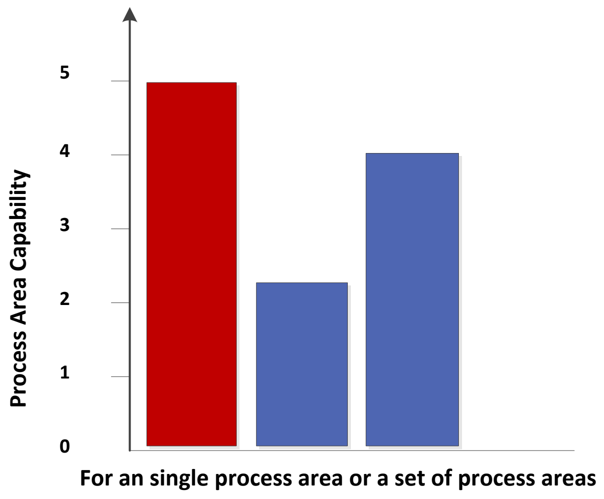
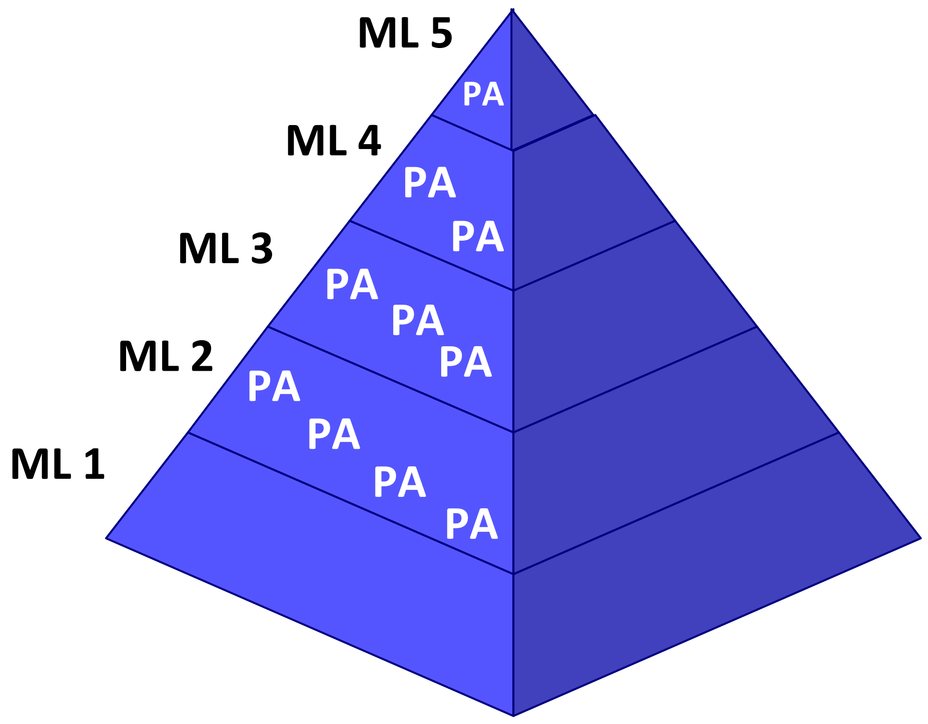


Figure 2 CMMI Continuous Presentation

The staged representation gives a standard series of improvements, and provides a foundation for the comparison the maturity of various organizations, and can serve as a basis for comparing the maturity of different projects and organizations. It is helpful in moving from CMM to CMMI [23].



For an established set of process areas of an organization

Figure 3 CMMI Staged Presentation

2.7.2. Levels of CMMI

The model can be applied to Development teams, Software projects, work group and the whole organizations all over the world. CMMI solutions use levels to recommend the desired evolutionary paths to the organizations that want to bring process improvement in developing and delivering the services. Both capability levels and maturity levels are supported by CMMI for Acquisition, development and services [12].

2.7.2.1. Capability Levels

A Capability level is a definite evolutionary level which illustrates an organization’s capability related to a process area. A capability level constitutes specific and generic practices for a process area that can improve the organization's processes associated with that process area [12].

Continuous representation of CMMI process areas, there are six capability levels from 0 to 5:

Level 0	Incomplete
Level 1	Performed
Level 2	Managed
Level 3	Defined
Level 4	Quantitatively Managed
Level 5	Optimizing

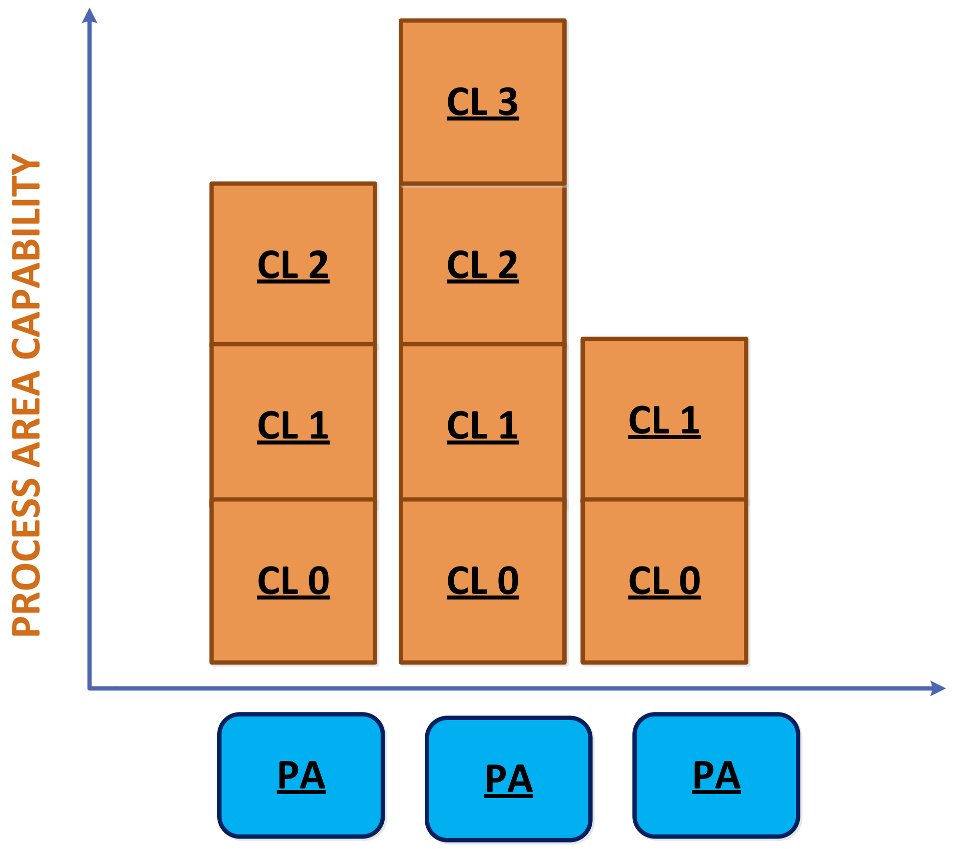


Figure 4 CMMI Capability levels [12]

2.7.2.2. Maturity Levels

CMMI is divided in 5 maturity levels that define the maturity of the organization:

Level 1	Initial
Level 2	Managed
Level 3	Defined
Level 4	Quantitatively managed
Level 5	Optimizing [12]

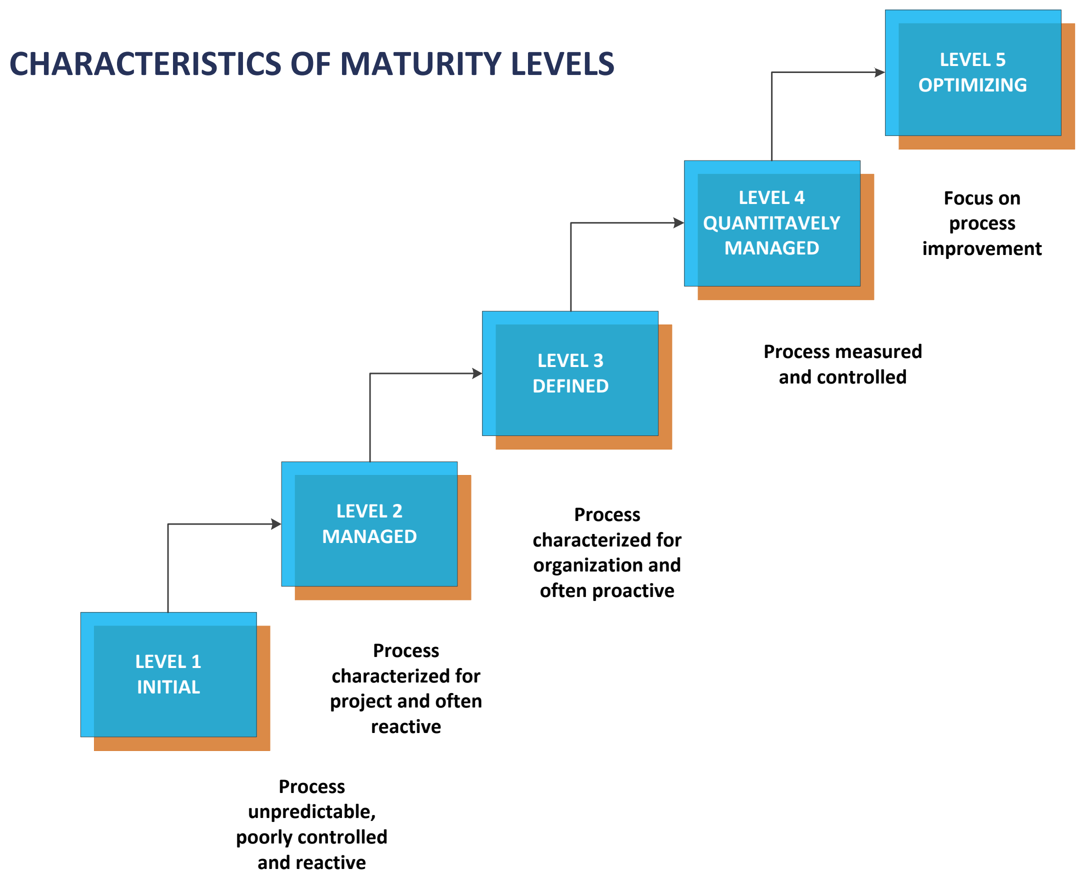

CHARACTERISTICS OF MATURITY LEVELS
LEVEL 5
OPTIMIZING
LEVEL 4
QUANTITAVELY
MANAGED
Focus on process improvement
LEVEL 3
DEFINED
Process measured and controlled
LEVEL 2
MANAGED
Process characterized for organization and often proactive
LEVEL 1
INITIAL
Process characterized for project and often reactive
Process unpredictable, poorly controlled and reactive

Figure 5 CMMI Maturity Levels [12]

CMMI model is composed of the best practices that help organizations in improving their efficiency and quality.

All CMMI models share the same:

i. **Architecture.** All CMMI models possess three parts:

a) Introduction

b) Process areas and generic goals and practices

c) Appendices.

CMMI models contain many process areas; every process area has a unique introduction, specific goals and practices.

ii. **Core process areas**. All CMMI models share 16 core process areas. These process areas are tailored for each model, but contain essentially the same information in each [12].

2.8. CMMI Model Framework

Because of the various versions and frameworks of CMMI for development, acquisition and service areas, SEI has introduced one framework that shares all the commonalities either of process areas or specific practices [12].

Some Process Areas, Specific Practices and generic Practices are common in Model for development, Model for Acquisition and Model for Services. And theses three models have their own dedicated process areas and amplifications [12].

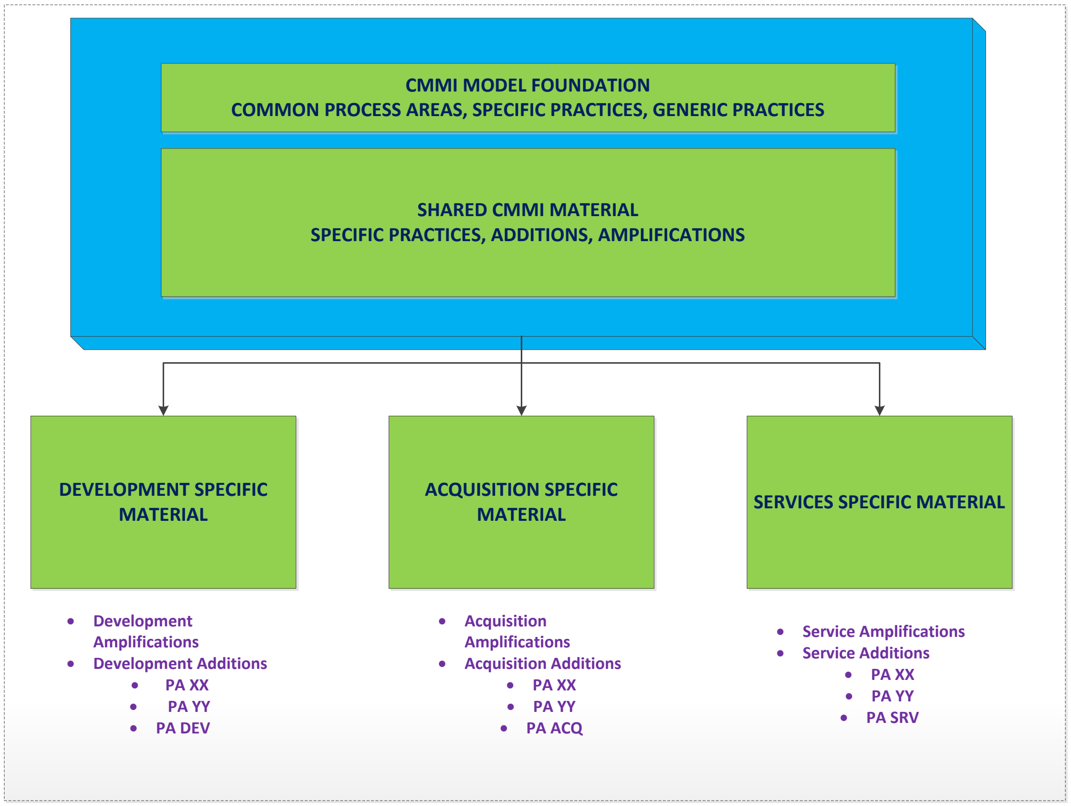

CMMI MODEL FOUNDATION
COMMON PROCESS AREAS, SPECIFIC PRACTICES, GENERIC PRACTICES
SHARED CMMI MATERIAL
SPECIFIC PRACTICES, ADDITIONS, AMPLIFICATIONS
DEVELOPMENT SPECIFIC MATERIAL
ACQUISITION SPECIFIC MATERIAL
SERVICES SPECIFIC MATERIAL
Development Amplifications
Development Additions
PA XX
PA YY
PA DEV
Acquisition Amplifications
Acquisition Additions
PA XX
PA YY
PA ACQ
Service Amplifications
Service Additions
PA XX
PA YY
PA SRV

Figure 6 CMMI Framework [12]

CMMI Model Foundation contains 16 Core Process Areas, CMMI models define a "process area" as a group of associated practices which are when carried out together, satisfies the criteria important for that area's improvement [12].

Process areas are:

ABBREVIATION	NAME	AREA	MATURITY LEVEL
CAR	Causal Analysis and Resolution	Support	5
CM	Configuration Management	Support	2
DAR	Decision Analysis and Resolution	Support	3
IPM	Integrated Project Management	Project Management	3
MA	Measurement and Analysis	Support	2
OPD	Organizational Process Definition	Process Management	3
OPF	Organizational Process Focus	Process Management	3
OPM	Organizational Performance Management	Process Management	5
OPP	Organizational Process Performance	Process Management	4
OT	Organizational Training	Process Management	3
PMC	Project Monitoring and Control	Project Management	2
PP	Project Planning	Project Management	2
PPQA	Process and Product Quality Assurance	Support	2
QPM	Quantitative Project Management	Project Management	4
REQM	Requirements Management	Project Management	2
RSKM	Risk Management	Project Management	3

Table 3 CMMI Core Process Areas [12]

Process areas found only in CMMI for Acquisition [25]:

1) Acquisition Requirements Development (ARD)

2) Solicitation and Supplier Agreement Development (SSAD)

3) Agreement Management (AM)

4) Acquisition Technical Management (ATM)

5) Acquisition Verification (AVER)

6) Acquisition Validation (AVAL)

Process found only in CMMI for Development [25]:

1) Product Integration (PI)

2) Requirements Development (RD)

3) Requirements Management (REQM)

4) Supplier Agreement Management (SAM)

5) Technical Solution (TS)

6) Validation (VAL)

7) Verification (VER)

Process areas found only in CMMI for Service [25]:

1) Capacity and Availability Management (CAM)

2) Incident Resolution and Prevention (IRP)

3) Supplier Agreement Management (SAM)

4) Service Continuity (SCON)

5) Service Delivery (SD)

6) Service System Development (SSD)

7) Service System Transition (SST)

8) Strategic Service Management (STSM)

2.8.1. Why CMMI

One problem being faced by Software industries is the selection of one Quality standard among all a number of SQA Standards that has been written and approved [12].

Why CMMI to be chosen, there are a number of reasons:

2.8.1.1. Business Success

CMMI had been beneficial for many business projects and Success stories have been shared by many business projects. The improvement has been witnessed in various areas like cost effectiveness, scheduling, return of investment, accuracy etc.

2.8.1.2. Cost Effective

CMMI is cost effective. All the benefits mentioned above can be acquired with return of investment.

2.8.1.3. Compatible

CMMI is compatible with all the other technologies such as Agile, Sigma, Scrum, ISO Standards, TSP etc.

2.8.1.4. Excellent Track Record

As mentioned already, many users have shared their success stories happened as a result of using CMMI. That's why CMMI has an excellent track record. Since its initiation in 1995, there is a large group of users who have been following CMMI. It is always improving. And at present too extensive research is ongoing.

2.9. CMMI LEVEL 2 (PPQA)

Process and product quality assurance is a Level 2 Process Area (PA) of Capability Maturity Model Integration (CMMI). Process and product quality assurance process area comprises of practices which make sure that the implementation of the planned processes has been done. Satisfying this PA is a major step towards achieving Level 2 of CMMI. This PA requires a written process for process and product quality assurance.

A Support process area at Maturity Level 2.

2.9.1. Purpose

The purpose of Process and Product Quality Assurance (PPQA) is to provide staff and management with objective insight into processes and associated work products.

2.9.2. Specific Practices by Goal

SG 1 Objectively Evaluate Processes and Work Products

SP 1.1 Objectively Evaluate Processes

SP 1.2 Objectively Evaluate Work Products and Services

SG 2 Provide Objective Insight

SP 2.1 Communicate and Ensure Resolution of Noncompliance Issues

SP 2.2 Establish Records [29]

Noncompliance issue is the mainly focused objective and one of the sub practices of "Establish records" are:

Subpractice 1: Activities of Process and product quality assurance should be recorded making sure that result and status of the project is clear.

2.10. SUMMARY

In this chapter, standard definitions of the terminologies related to Quality, Software Quality, Software Quality standards and CMMI have been presented. The background study on CMMI and its process areas was discussed. These details will help reader in better understanding of the proposed approach in the following chapter.

Chapter 3

RELATED WORK

3.1. INTRODUCTION

This chapter provides an insight to the existing research that is done to either Quality Assurance or CMMI. It illustrates different methods, frameworks, techniques and models proposed by researcher to enhance efficiency and accuracy of CMMI framework.

3.2. RELATED WORK

In [14], Rubey and Brewer has described the various Software Quality Assurance standards that exist, along with the comparisons among them. All the standards described are then combined forming one generic Quality Assurance Standard. This Generic standard may help the organizations in achieving Quality Assurance based on all the Quality Assurance standards of the World.

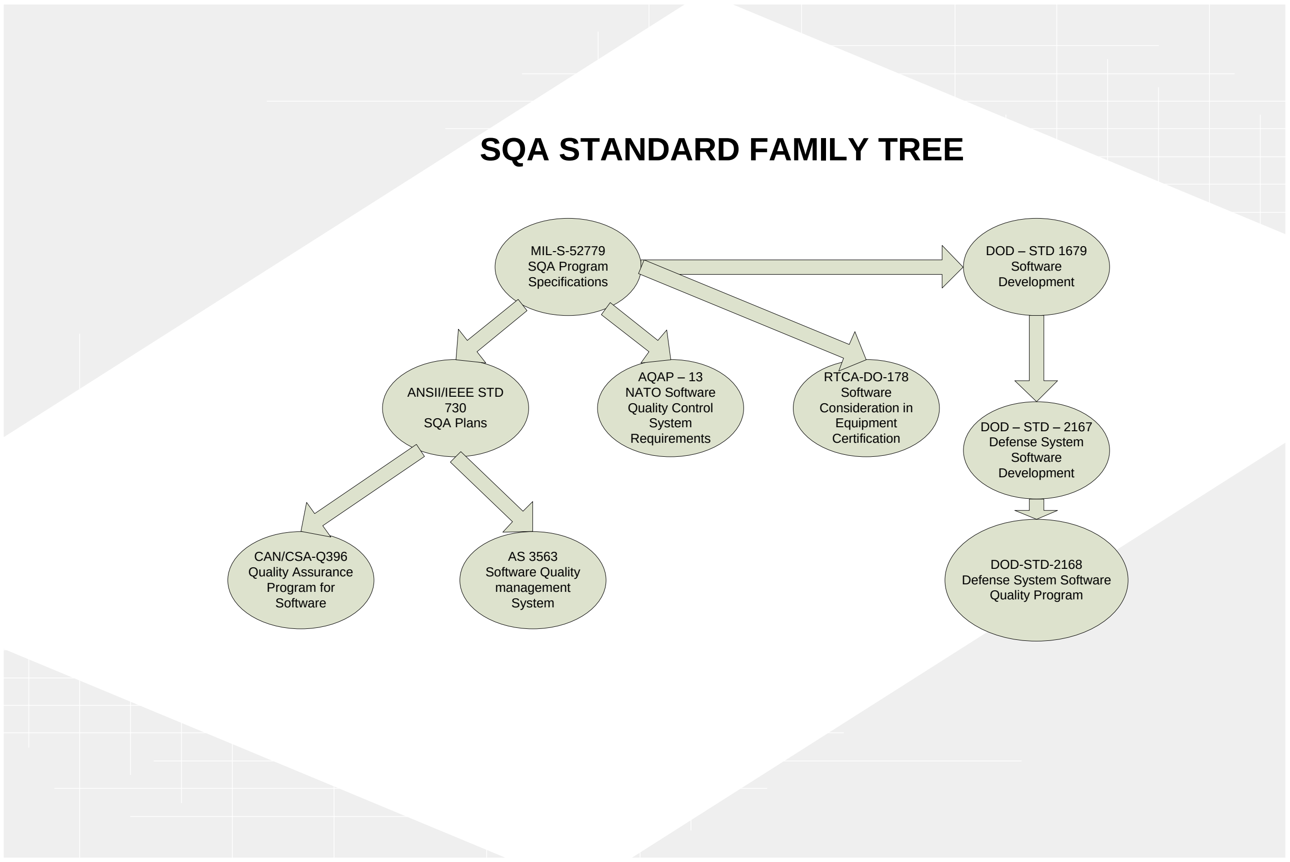

SQA STANDARD FAMILY TREE
MIL-S-52779 SQA Program Specifications
DOD – STD 1679 Software Development
ANSII/IEEE STD 730 SQA Plans
AQAP – 13 NATO Software Quality Control System Requirements
RTCA-DO-178 Software Consideration in Equipment Certification
DOD – STD – 2167 Defense System Software Development
CAN/CSA-Q396 Quality Assurance Program for Software
AS 3563 Software Quality management System
DOD-STD-2168 Defense System Software Quality Program

Figure 7 Software Quality Assurance Standards Family Tree

In [15], Wells, Brand and Markosian described a new approach presenting a toolset used for measuring source code compliance with design and coding standards and also performing quality assurance for new applications.

In [16], Serrano, Carlos and Cedillo shared the experience of using a software process for implementing Software Process Improvement in a software development company.

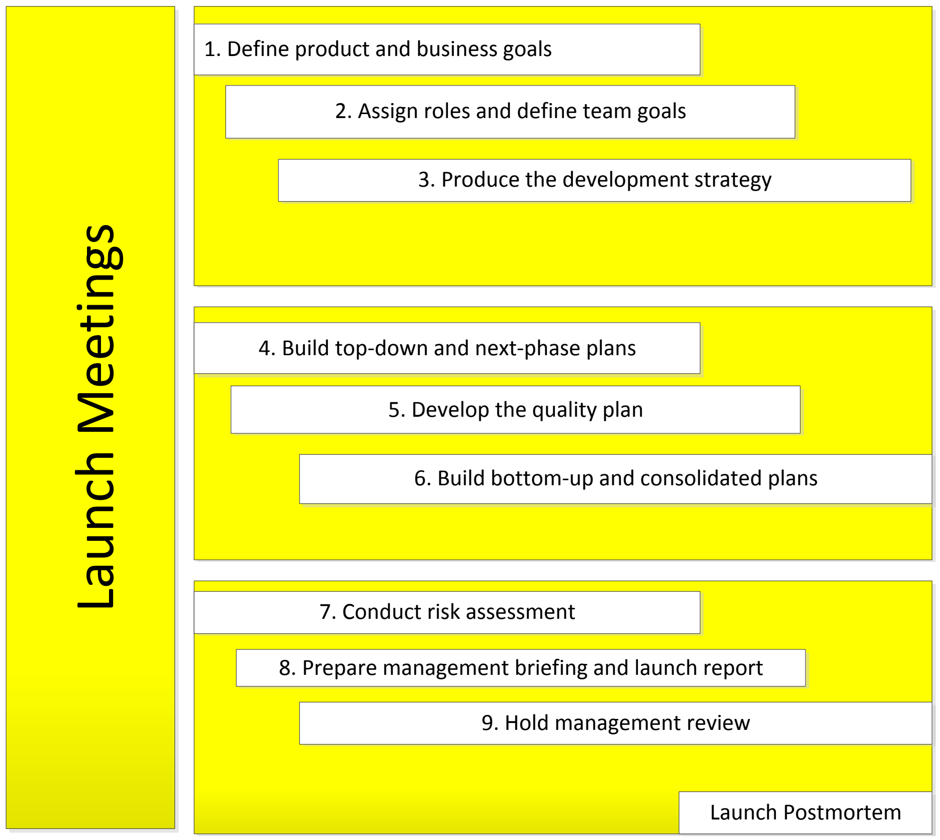

Launch Meetings
1. Define product and business goals
2. Assign roles and define team goals
3. Produce the development strategy
4. Build top-down and next-phase plans
5. Develop the quality plan
6. Build bottom-up and consolidated plans
7. Conduct risk assessment
8. Prepare management briefing and launch report
9. Hold management review
Launch Postmortem

Figure 8 TSP Launch Structure [16]

The proposal of implementing a software improvement process based on family of products, including a model (CMM), an implementation model (IDEAL), and assessment methodm(CBA-IP) and specific processes (TSP and PSP), turns out to be a success clearing that TSP helps in adapting the process oriented culture.

In [17], Amaral and Faria proposed a methodology for assessing the effectiveness of Team Software Process that was created by SEI (Software Engineering Institute). Thismethodology is helpful in the assessment of future gains that an organization will confer during and after the TSP is implemented.

Following research methodology was adapted:

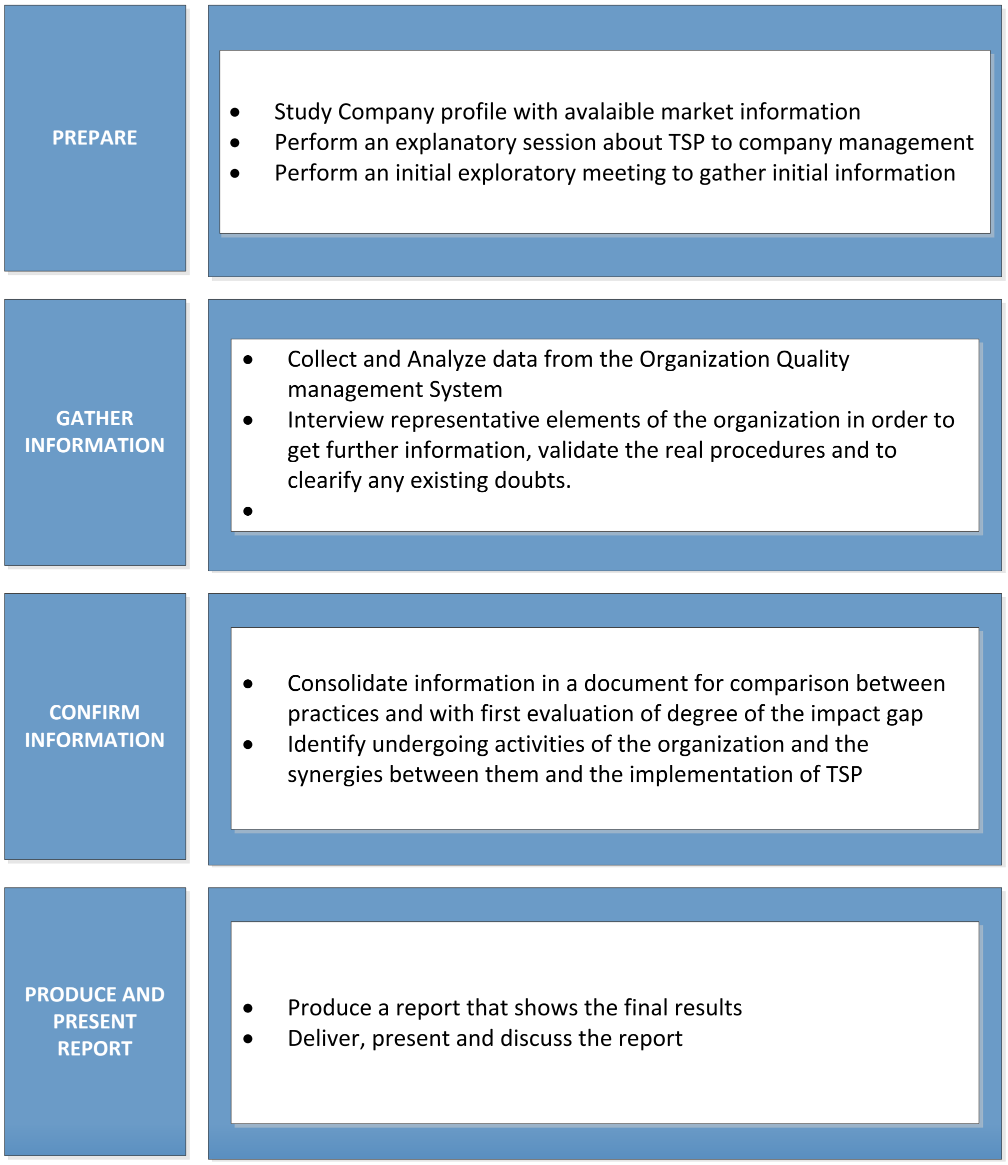

PREPARE
Study Company profile with avalaible market information
Perform an explanatory session about TSP to company management
Perform an initial exploratory meeting to gather initial information
GATHER INFORMATION
Collect and Analyze data from the Organization Quality management System
Interview representative elements of the organization in order to get further information, validate the real procedures and to clearify any existing doubts.
CONFIRM INFORMATION
Consolidate information in a document for comparison between practices and with first evaluation of degree of the impact gap
Identify undergoing activities of the organization and the synergies between them and the implementation of TSP
PRODUCE AND PRESENT REPORT
Produce a report that shows the final results
Deliver, present and discuss the report

Figure 9 TSP Main passes and activities in the proposed Gap Analysis methodology [17]

The proposed methodology gives the status of Gap before the project begins. Its appliaction gives an organization more distributed internal effort because of its traits i.e cost effective and faster performance.

In [18], Karg, Grottke and Beckhaus has broadened the limited work done on empirical knowledge in software quality cost research area.

They have modulated the function as:

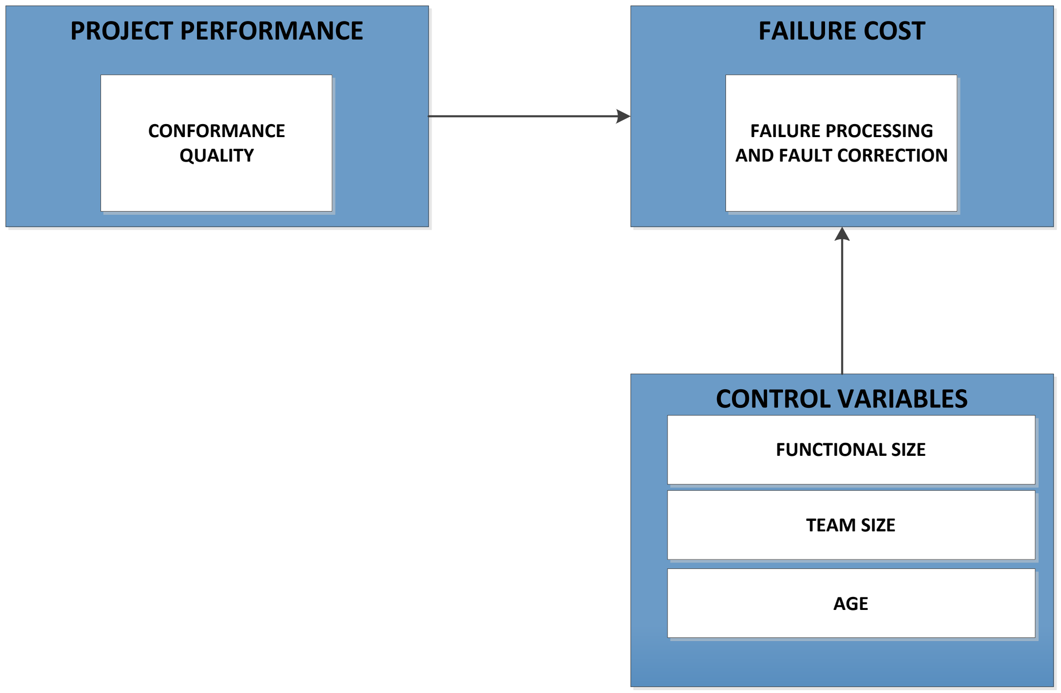


Figure 10 TSP Conceptual Framework [18]

In [19], Otte, Moreton, Knoell developed a QA framework which is capable of suggesting product quality targets and QA processes. The requirements of the developed framework are balanced Human interaction factors, feasible environment to achieve software quality and managerial skills.

The generic process framework of the developed framework is as follows:

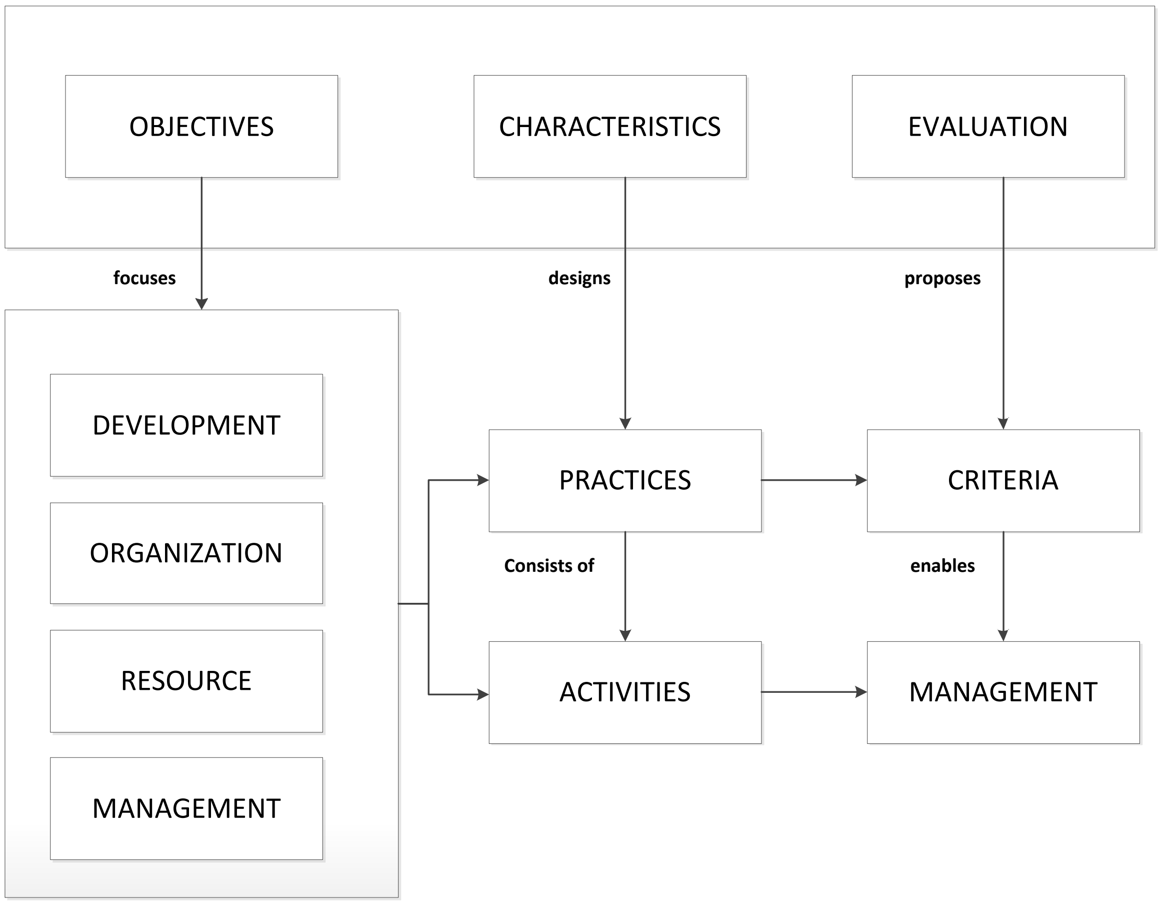


Figure 11 Generic Process Framework [19]

The framework is formed as a reference model which can be tailored and it put together all those process areas together that play important role in Software Quality Assurance.

In [20], Alsultanny and Wohaishi have suggested a model with the help of which the factors affecting the software quality and software productivity can be tested. The proposed model helps in reaching Quality standard 9126 by taking all factors affecting Software quality.

In [21], Niazi and Babar presented an empirical study with the aim of finding the CMMI Level 2 Specific practices' "perceived value" keeping the experiences of developers of small sized companies. The main focus is to find out the level to which a CMMI practice makes its contribution in the development of finer grained framework.

3.3. SUMMARY

This chapter presented the previous related work about Quality Assurance and CMMI. The presented concepts stand a reason for this thesis. It described several models, methodologies and frameworks that are proposed by researches that help in achieving Quality assurance in a better way. Nevertheless, No model was still proposed that may help CMMI Level 2 Process and Product Quality Area to be implemented in a better and effective way.

Chapter 4

PROPOSED APPROACH

4.1. INTRODUCTION

To propose a Model for CMMI level 2 Process and Product Quality Assurance Process area which may support in Quality related objectives of CMMI Level 2 Software project for compliance, an empirical study of a CMMI level 2 Software project is carried out. Based on the knowledge gained from the study, a Model is evaluated. The proposed model stands as a guideline for defining a Quality assurance program for a CMMI level 2 certified company projects.

Designing a process for quality assurance will facilitate organizational strategy toward process and project improvements. For the sake of implementation, based on the design of the model, a tool named QAGA (Quality Assurance Gap Analyzer) has been developed. QAGA works on the approach of the proposed models. It takes requirements and test cases as inputs and generates a report giving information about the Gap that exists in the coverage of the Test cases which is no doubt an aiding step towards the Quality Assurance of the Software product.

The sequence of the research methodology is as follows:

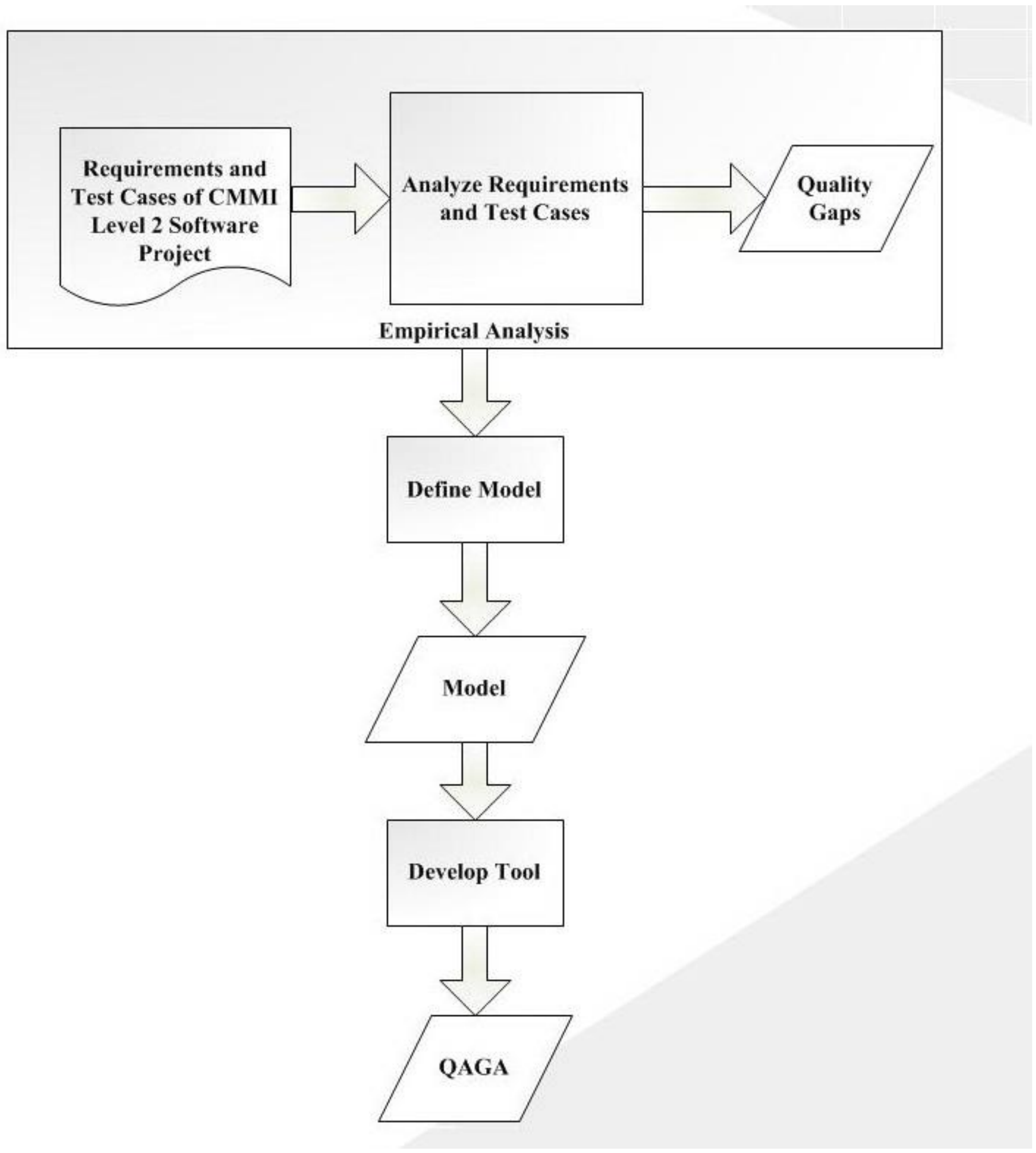
i. Literature review of CMMI Key Process Areas

ii. Empirical study of CMMI level 2 Software Project

iii. Designing the Model

iv. Developing the tool (QAGA)

v. Results and Analysis



Requirements and Test Cases of CMMI Level 2 Software Project
Analyze Requirements and Test Cases
Quality Gaps
Empirical Analysis
Define Model
Model
Develop Tool
QAGA

Figure 12 Research Methodology

This chapter presents the details of the each step that is carried out mentioned above.

4.2. LITERATURE REVIEW OF CMMI KEY PROCESS AREAS

Literature Review of the CMMI and its Process areas is done firstly. The Quality standards proposed till time has been studied. Different quality frameworks developed by various researchers are also reviewed.

4.3. EMPIRICAL STUDY OF CMMI LEVEL 2 SOFTWARE PROJECT

The CMMI Level 2 Software Project for empirical study was taken from a Private Software Development Organization which deals in Software Development focusing on Communication Systems and Information Technology and its one of the core competences is software design. Organization is CMMI Level 2 certified. The Software project was empirically studied in a series of steps which are as follows:

i. Requirement Analysis

ii. Test case Analysis

iii. Mapping Requirements and Test Cases

iv. Calculating the Missing Gaps

v. Gap Analysis Results

4.3.1. Requirement Analysis

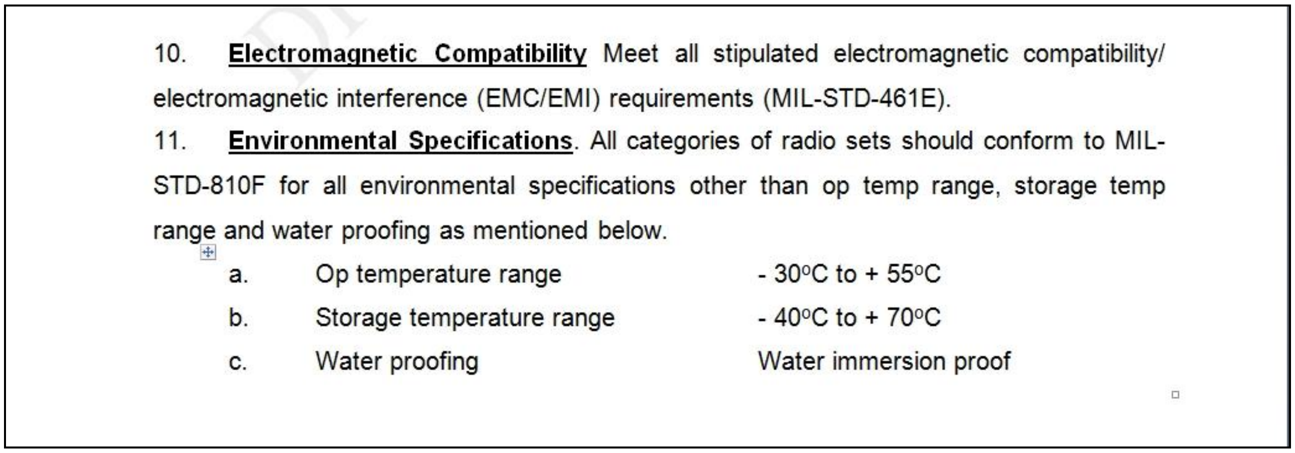
First phase of empirical study is “Requirement Analysis”. The requirements of the software project are studied thoroughly and thus analyzed accordingly.

The Requirement Analysis is done in the following series of steps:

i. Studying each Requirement clause stated in the GSR document
ii. Categorizing the requirement i.e. General, Technical, environmental or Auxiliary.
iii. Assigning a Unique ID to each Requirement
iv. Defining the Criticality of each Requirement
v. Defining the Dependency of each Requirement

4.3.1.1. Studying each Requirement clause

GSR (General Staff requirement) is the document containing all the requirement clauses that are specified by the client to the developer. GSR document of the CMMI level 2 software project is obtained from the organization. The GSR document contains different requirement clauses and each requirement clause is studied thoroughly.



10. **Electromagnetic Compatibility** Meet all stipulated electromagnetic compatibility/ electromagnetic interference (EMC/EMI) requirements (MIL-STD-461E).

11. **Environmental Specifications**. All categories of radio sets should conform to MIL-STD-810F for all environmental specifications other than op temp range, storage temp range and water proofing as mentioned below.

a.	Op temperature range	- 30°C to + 55°C
b.	Storage temperature range	- 40°C to + 70°C
c.	Water proofing	Water immersion proof

Figure 13 Requirement Clauses in a GSR Document

4.3.1.2. Categorizing the requirement

Once requirements are studied, all are listed down naming the categories they fall in. Categories include:

i. General Requirements

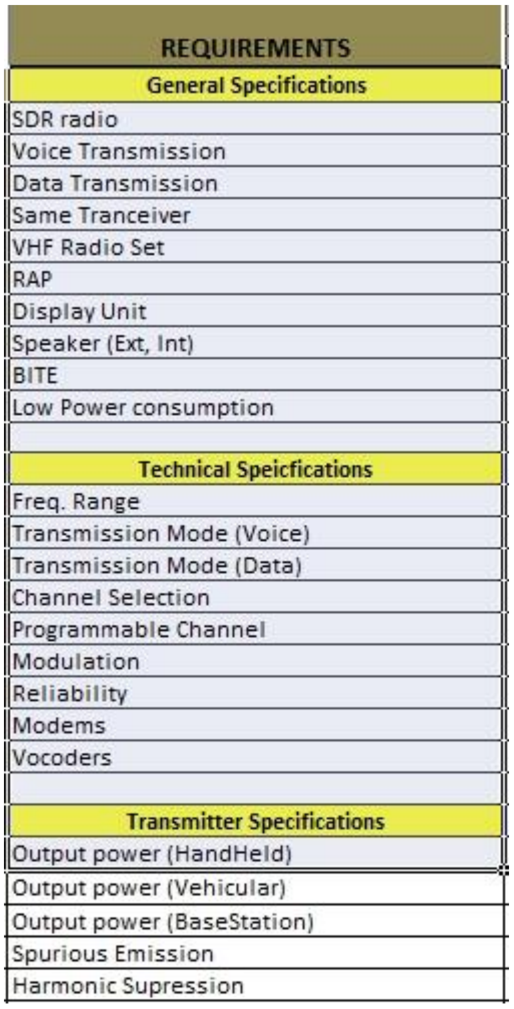
ii. Technical Specifications

iii. Power Specifications

iv. Special features Specifications

v. Environmental Specifications

vi. Auxiliaries

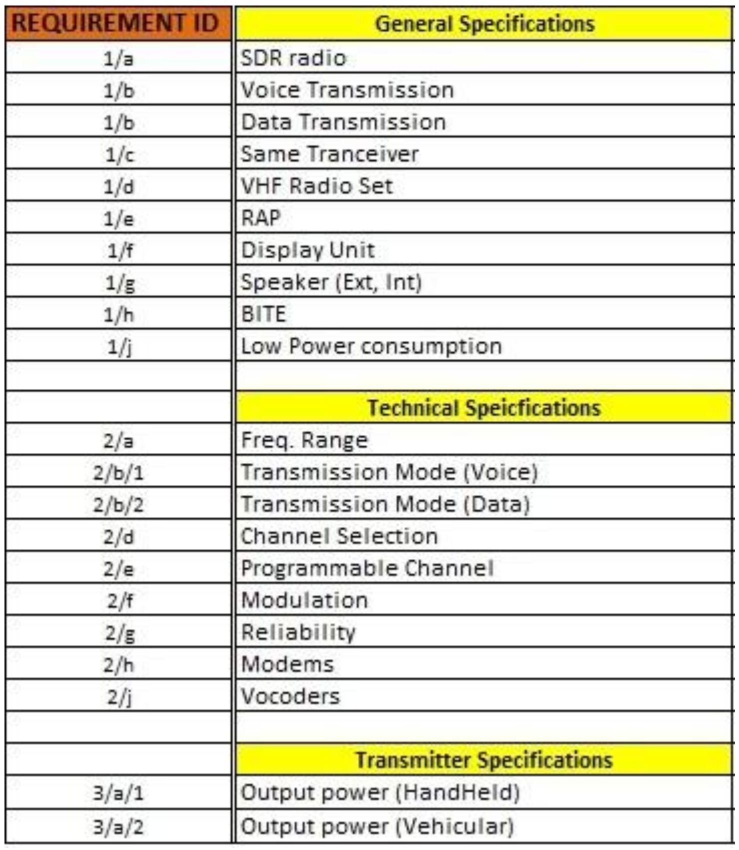


REQUIREMENTS
General Specifications
SDR radio
Voice Transmission
Data Transmission
Same Tranceiver
VHF Radio Set
RAP
Display Unit
Speaker (Ext, Int)
BITE
Low Power consumption
Technical Speicfications
Freq. Range
Transmission Mode (Voice)
Transmission Mode (Data)
Channel Selection
Programmable Channel
Modulation
Reliability
Modems
Vocoders
Transmitter Specifications
Output power (HandHeld)
Output power (Vehicular)
Output power (BaseStation)
Spurious Emission
Harmonic Supression

Figure 14 Requirement distributed in categories

4.3.1.3. Assigning a Unique ID

For the unique identification, a unique ID is given to each requirement.



REQUIREMENT ID	General Specifications
1/a	SDR radio
1/b	Voice Transmission
1/b	Data Transmission
1/c	Same Tranceiver
1/d	VHF Radio Set
1/e	RAP
1/f	Display Unit
1/g	Speaker (Ext, Int)
1/h	BITE
1/j	Low Power consumption
	Technical Speicfications
2/a	Freq. Range
2/b/1	Transmission Mode (Voice)
2/b/2	Transmission Mode (Data)
2/d	Channel Selection
2/e	Programmable Channel
2/f	Modulation
2/g	Reliability
2/h	Modems
2/j	Vocoders
	Transmitter Specifications
3/a/1	Output power (HandHeld)
3/a/2	Output power (Vehicular)

Figure 15 Requirements assigned unique IDs

4.3.1.4. Defining the Criticality

The criticality of the requirement by the following characteristics:

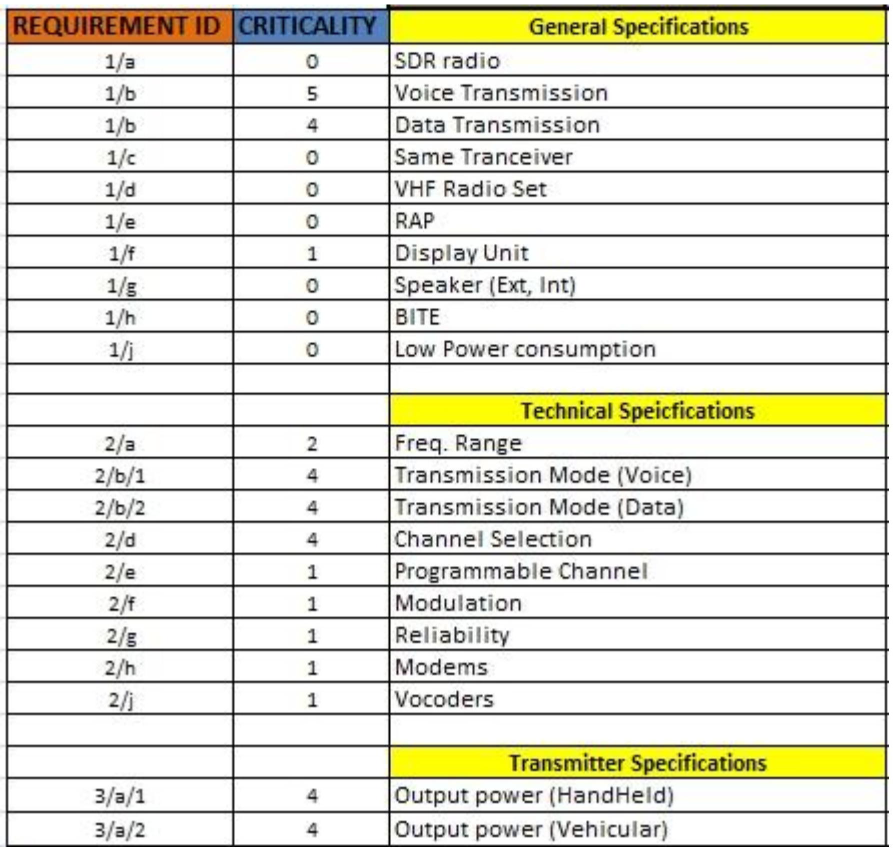
i. Address several test cases

ii. Has a high level of control

iii. Is complex or error prone

iv. Has definite performance specifications [22]

The criticality range is from 0 to 5, where “0” is the least critical and “5” is the most critical Requirement.



REQUIREMENT ID	CRITICALITY	General Specifications
1/a	0	SDR radio
1/b	5	Voice Transmission
1/b	4	Data Transmission
1/c	0	Same Tranceiver
1/d	0	VHF Radio Set
1/e	0	RAP
1/f	1	Display Unit
1/g	0	Speaker (Ext, Int)
1/h	0	BITE
1/j	0	Low Power consumption
		Technical Speicfications
2/a	2	Freq. Range
2/b/1	4	Transmission Mode (Voice)
2/b/2	4	Transmission Mode (Data)
2/d	4	Channel Selection
2/e	1	Programmable Channel
2/f	1	Modulation
2/g	1	Reliability
2/h	1	Modems
2/j	1	Vocoders
		Transmitter Specifications
3/a/1	4	Output power (HandHeld)
3/a/2	4	Output power (Vehicular)

Figure 16 Requirement and their Criticalities

4.3.1.5. Defining the Dependency

The dependency among requirements also exists. A Requirement is dependent on the other if a test case dedicated to that requirement cannot be fulfilled because the same test case is being used in some other requirement and working as an input in the second one.

4.3.2. Test Cases Analysis

For the purpose of Test case analysis, Test cases Document of the same CMMI Level 2 Software project is studied. Test case Analysis is done in the following steps:

i. Studying the Test case Document thoroughly

ii. Assigning a unique ID to each Test case

4.3.2.1. Studying the Test Case Document

Test Case Document contains all the Test cases that are performed for the complete testing of a Software Project. Test Case Document list down all the following salient information about testing the project.

i. Introduction

ii. Scope

iii. Test and measurement Principles

iv. Test Equipment List

v. Inspection and Test Procedures

vi. Specification tests

vii. Specification Test Procedures

viii. Test Record tables

ix. Change Records

Test case Document, with respect to this very project contains the test cases addressing all of its requirements like:

i. Compatibility tests

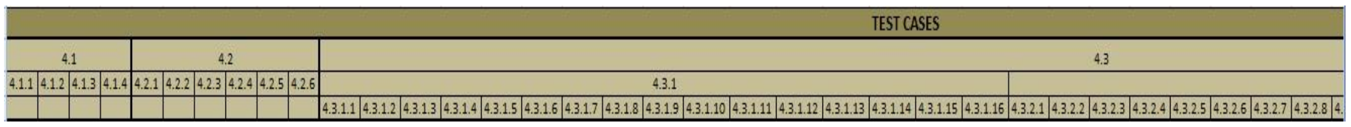
ii. General specification tests

iii. Electrical tests

iv. Environmental Test

4.3.2.2. Assigning a unique Test case ID

Each Test Case in the Test case document is assigned a unique ID.



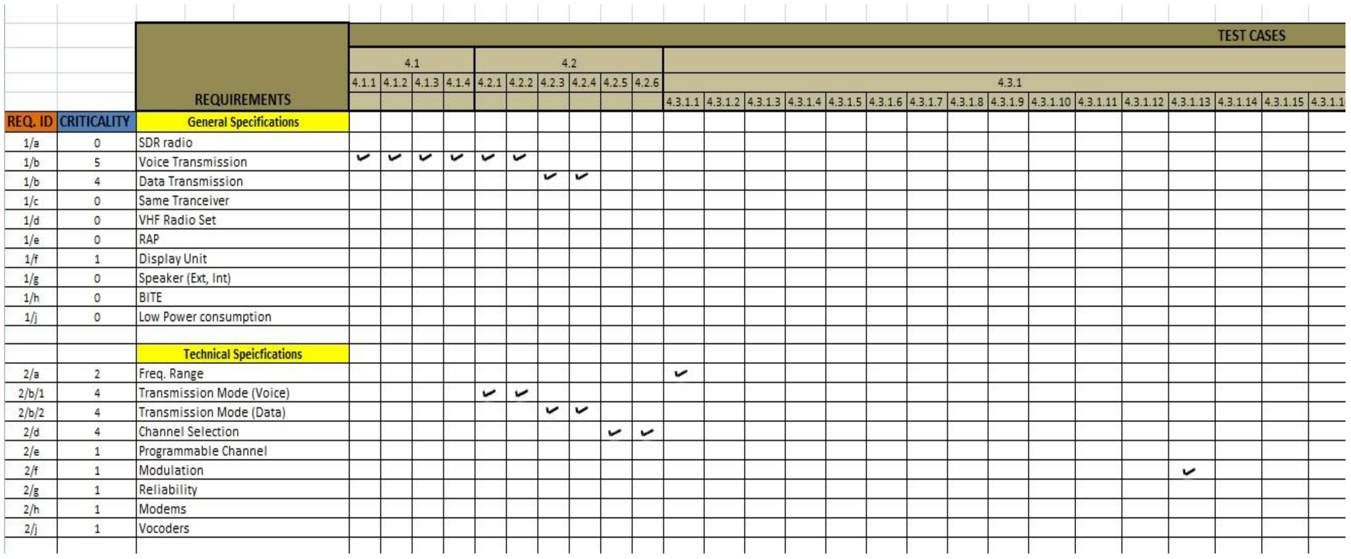
TEST CASES																																		
4.1				4.2						4.3																								
4.1.1	4.1.2	4.1.3	4.1.4	4.2.1	4.2.2	4.2.3	4.2.4	4.2.5	4.2.6	4.3.1																								
										4.3.1.1	4.3.1.2	4.3.1.3	4.3.1.4	4.3.1.5	4.3.1.6	4.3.1.7	4.3.1.8	4.3.1.9	4.3.1.10	4.3.1.11	4.3.1.12	4.3.1.13	4.3.1.14	4.3.1.15	4.3.1.16	4.3.2.1	4.3.2.2	4.3.2.3	4.3.2.4	4.3.2.5	4.3.2.6	4.3.2.7	4.3.2.8	4.

Figure 17 Test cases with Unique IDs

4.3.3. Mapping Requirements and Test Cases

Once all the Requirements are categorized assigning unique IDs to each Requirement, and also all the Test Cases are given unique IDs. The next step is to make a metric plotting all the Requirements along with the Test Cases.

Following is the view of the Metric developed by plotting all Requirements of CMMI Level 2 Software project against all the Test Cases.



			TEST CASES																									
			4.1				4.2																					
													4.3.1															
		REQUIREMENTS	4.1.1	4.1.2	4.1.3	4.1.4	4.2.1	4.2.2	4.2.3	4.2.4	4.2.5	4.2.6	4.3.1.1	4.3.1.2	4.3.1.3	4.3.1.4	4.3.1.5	4.3.1.6	4.3.1.7	4.3.1.8	4.3.1.9	4.3.1.10	4.3.1.11	4.3.1.12	4.3.1.13	4.3.1.14	4.3.1.15	4.3.1.1
REQ. ID	CRITICALITY	General Specifications																										
1/a	0	SDR radio																										
1/b	5	Voice Transmission	✓	✓	✓	✓	✓	✓																				
1/b	4	Data Transmission							✓	✓																		
1/c	0	Same Tranceiver																										
1/d	0	VHF Radio Set																										
1/e	0	RAP																										
1/f	1	Display Unit																										
1/g	0	Speaker (Ext, Int)																										
1/h	0	BITE																										
1/j	0	Low Power consumption																										
		Technical Speicfications																										
2/a	2	Freq. Range											✓															
2/b/1	4	Transmission Mode (Voice)					✓	✓																				
2/b/2	4	Transmission Mode (Data)							✓	✓																		
2/d	4	Channel Selection									✓	✓																
2/e	1	Programmable Channel																										
2/f	1	Modulation																							✓			
2/g	1	Reliability																										
2/h	1	Modems																										
2/j	1	Vocoders																										

Figure 18 Mapping Requirements against test Cases

4.3.4. Assigning weights to the Test Cases

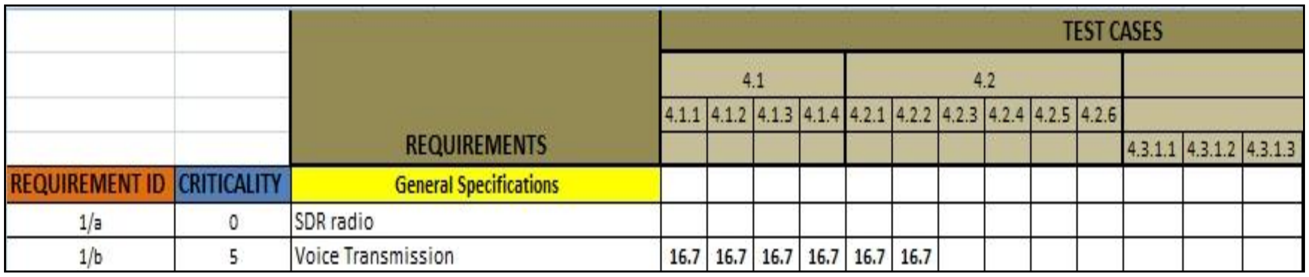
Each test case is assigned a particular weight according to its contribution in testing of that very requirement. The total weight age of the test cases for a requirement adds upto 100%.

Suppose R_1T_1 and R_1T_2 are the two Test cases contributes in the testing of Requirement R_1. So the weights of the Test Cases are assigned as:

$$R_1T_1 + R_1T_2 = 100\% \qquad \text{(i)}$$

So Weight of Test Case $R_1T_1 = 50$

And Weight of Test Case $R_1T_2 = 50$

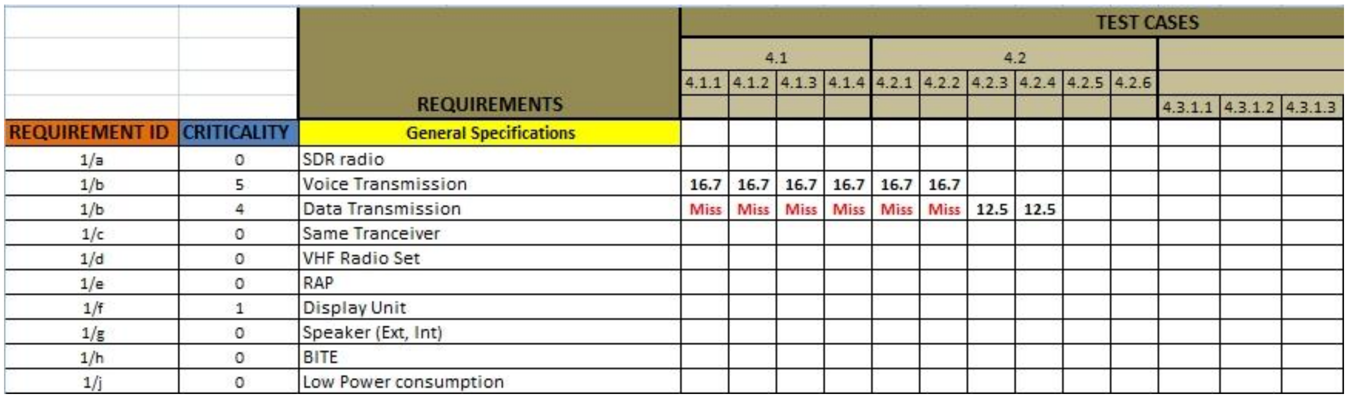


		REQUIREMENTS	TEST CASES												
			4.1				4.2								
			4.1.1	4.1.2	4.1.3	4.1.4	4.2.1	4.2.2	4.2.3	4.2.4	4.2.5	4.2.6	4.3.1.1	4.3.1.2	4.3.1.3
REQUIREMENT ID	CRITICALITY	General Specifications													
1/a	0	SDR radio													
1/b	5	Voice Transmission	16.7	16.7	16.7	16.7	16.7	16.7							

Figure 19 Assigning weights to Test Cases

4.3.5. Calculating the Missing gaps

Once all the test cases are assigned the weights according to their contribution in testing a requirement, the missing Test cases are highlighted.

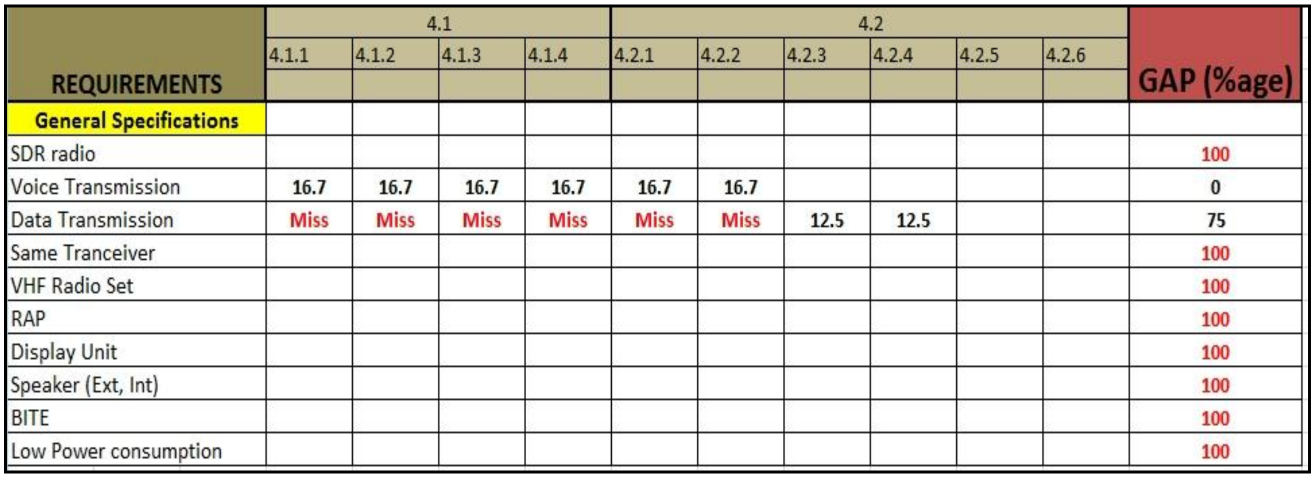


		REQUIREMENTS	TEST CASES												
			4.1				4.2								
			4.1.1	4.1.2	4.1.3	4.1.4	4.2.1	4.2.2	4.2.3	4.2.4	4.2.5	4.2.6	4.3.1.1	4.3.1.2	4.3.1.3
REQUIREMENT ID	CRITICALITY	General Specifications													
1/a	0	SDR radio													
1/b	5	Voice Transmission	16.7	16.7	16.7	16.7	16.7	16.7							
1/b	4	Data Transmission	Miss	Miss	Miss	Miss	Miss	Miss	12.5	12.5					
1/c	0	Same Tranceiver													
1/d	0	VHF Radio Set													
1/e	0	RAP													
1/f	1	Display Unit													
1/g	0	Speaker (Ext, Int)													
1/h	0	BITE													
1/j	0	Low Power consumption													

Figure 20 Calculating Missing Gaps

4.3.6. Gap Analysis Results

Once all the missing and completed Test cases are listed down. There comes the stage of calculating the Gap in testing each requirement. After calculating Gap of each requirement, an overall Average Gap of the Software Project is calculated. This is achieved by taking the average of the Gaps of all the Requirements of the Project.



REQUIREMENTS	4.1				4.2						GAP (%age)
	4.1.1	4.1.2	4.1.3	4.1.4	4.2.1	4.2.2	4.2.3	4.2.4	4.2.5	4.2.6	
General Specifications											
SDR radio											100
Voice Transmission	**16.7**	**16.7**	**16.7**	**16.7**	**16.7**	**16.7**					**0**
Data Transmission	**Miss**	**Miss**	**Miss**	**Miss**	**Miss**	**Miss**	**12.5**	**12.5**			**75**
Same Tranceiver											100
VHF Radio Set											100
RAP											100
Display Unit											100
Speaker (Ext, Int)											100
BITE											100
Low Power consumption											100

Figure 21 Total Gap Analysis

4.4. DESIGNING THE MODEL

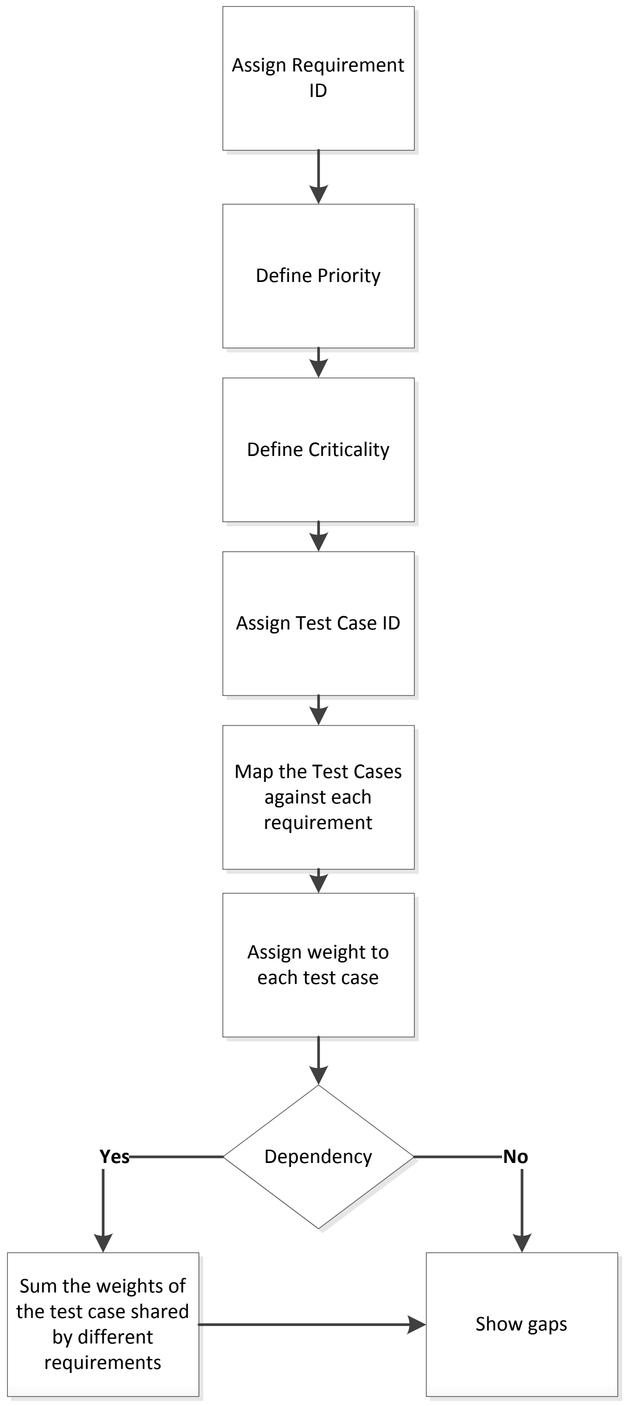

Assign Requirement ID
Define Priority
Define Criticality
Assign Test Case ID
Map the Test Cases against each requirement
Assign weight to each test case
Dependency
Yes
No
Sum the weights of the test case shared by different requirements
Show gaps

Figure 22 The Model

4.5. DEVELOPING THE TOOL

Based on the knowledge gained from the Empirical Study and the Proposed Model, a tool has been generated using C sharp and MS Access which is responsible for calculating Gaps of the Requirements of the Software Project and an overall Average Gap of the whole Software. The complete description of the Tool development along with its Results in discussed in chapter 5.

4.6. SUMMARY

In this chapter the research methodology is discussed in detail. The detail of the literature review of the area was presented. Then the steps taken to carry out the empirical study are elaborated. Then, the proposed Model on the knowledge acquired from empirical study has been discussed.

Finally, the tool developed taking the Model design as base, is presented.

Chapter 5

IMPLEMENTATION and RESULTS

5.1. INTRODUCTION

Quality Assurance Gap Analyzer (QAGA) is a comprehensive tool that addresses the coverage of Test Cases built for testing the Requirements of a Software project so that the results can be evaluated for the measurement of Quality Assurance of the Project. It is developed on the idea of the proposed Model in Chapter 4.

The QAGA tool takes Software Requirements and Test Cases as Inputs and gives an output of the average calculated Gap in the Quality of the Software Project, hence standing as a guideline in achieving CMMI level 2 (Process and Product Quality Assurance).

5.2. QAGA (QUALITY ASSURANCE GAP ANALYZER)

Implementation of the proposed model is done in the form of developing a tool, QAGA (Quality Assurance Gap Analyzer). The language used in its development is C sharp, and the Database tool used is MS Access.

5.3. REQUIREMENT As INPUT

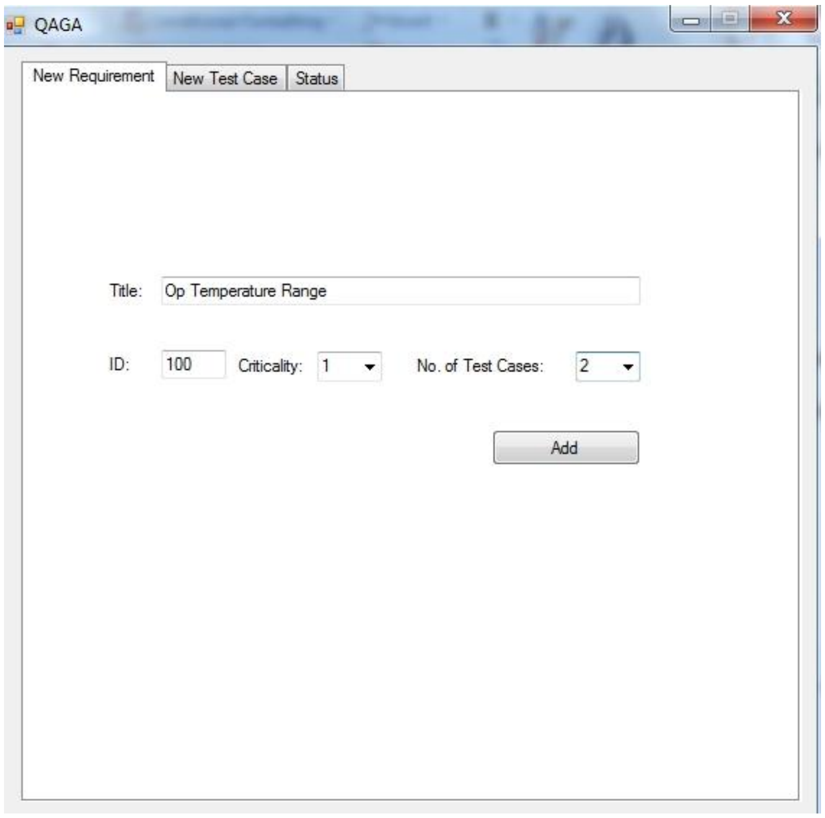
QAGA takes requirements as input one at a time. Following is the interface for the requirement input.

i. Title of the Requirement is entered.

ii. Next, ID of the Requirement is entered.

iii. Next, the criticality of the Requirement is entered.

iv. Finally, the number of Test cases that fulfill the very requirement is entered.



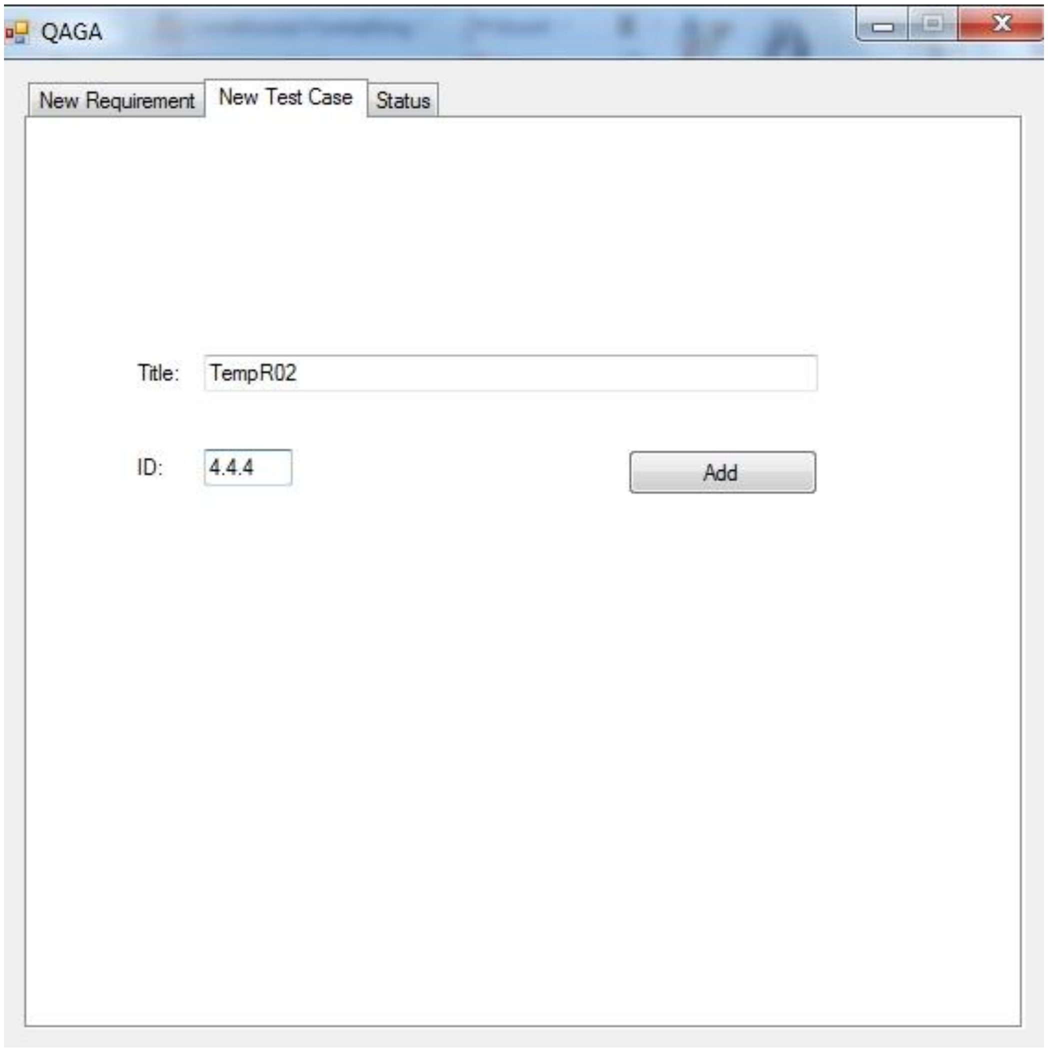
Title: Op Temperature Range

ID: 100 Criticality: 1 No. of Test Cases: 2

Figure 23 Requirement as Input

5.4. TEST CASES As INPUT

Once the requirement is entered, next step is to enter the test cases that are related to that requirement.



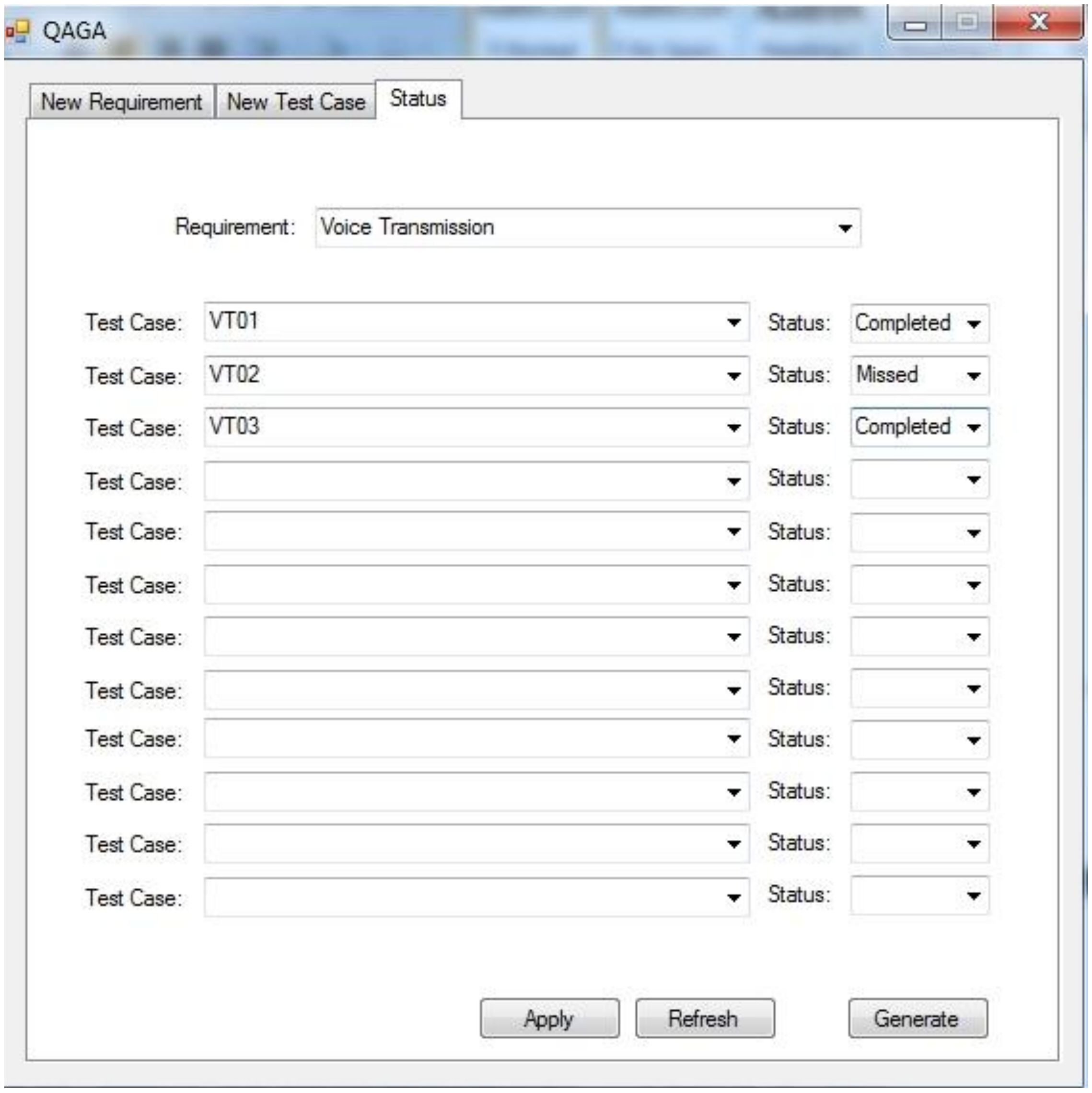
Title: TempR02

ID: 4.4.4

Figure 24 Test cases as Input

5.5. MAPPING OF REQUIREMENTS and TEST CASES

Status of each test case is against a particular requirement is entered. The status of the Test Case is either Completed or Missed. Based on the knowledge of the status, QAGA has the ability to calculate the weight of the missing Gaps in the Testing Phase.



Requirement: Voice Transmission

Test Case:	VT01	Status:	Completed
Test Case:	VT02	Status:	Missed
Test Case:	VT03	Status:	Completed
Test Case:		Status:	
Test Case:		Status:	
Test Case:		Status:	
Test Case:		Status:	
Test Case:		Status:	
Test Case:		Status:	
Test Case:		Status:	
Test Case:		Status:	
Test Case:		Status:	

Figure 25 Mapping of Requirements and Test Cases

5.6. GAP ANALYSIS RESULTS

QAGA takes the Requirement and all Test cases related to that requirement as Input and generate a Report giving complete information of Requirement Titles, Requirement IDs, its criticality, number of associated Test cases, Test Case Titles and Test Case IDs along with the Individual GAP in a particular requirement and an overall GAP in the whole Software Project taking the average of all individual Gaps. Also it plots a graph about the status of Gap of each requirement of the Software Project. This information is helpful in knowing that how far is the completion of testing of whole software project.

5.6.1. Scenario 01

Talking of a requirement named "Voice Transmission". None of the associated Test case is executed or in simple words all the Test cases are missed. The Result generated by QAGA for such a requirement is as follows:

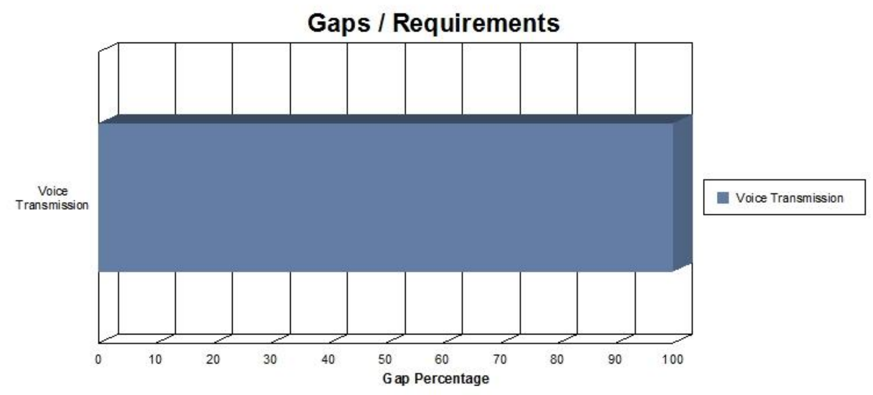


Figure 26 Scenario 01: GAP is 100%

The GAP of the Requirement “Voice Transmission” is 100% as no Testing of the requirement has been performed.

5.6.2. Scenario 02

If the same Requirement as described in Scenario 01 is taken again and all the test cases associated to it are performed successfully. Then QAGA shows the output as:

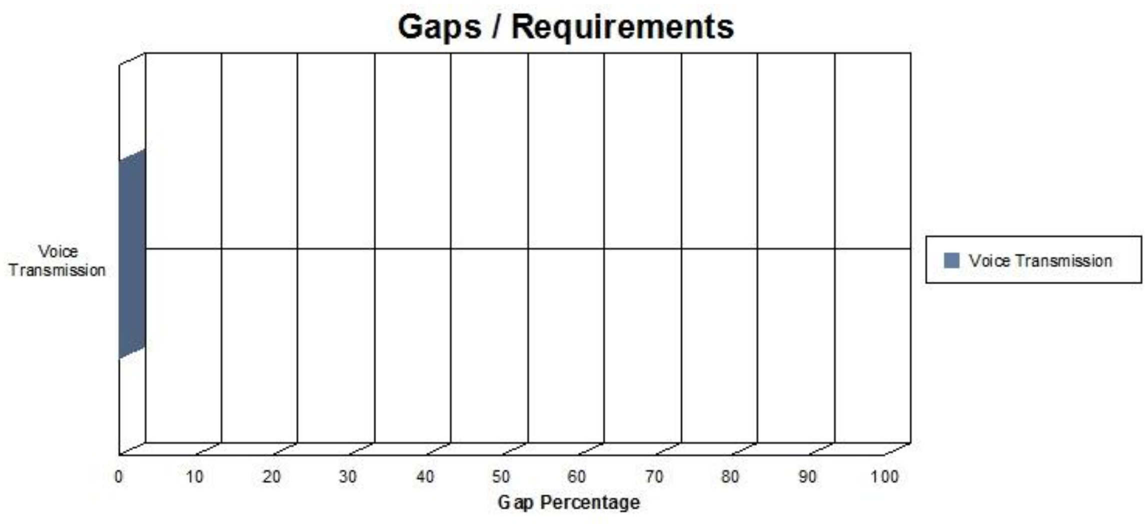


Figure 27 Scenario 02: GAP is 0%

As all the Test Cases related to the Requirement have been performed successfully, so the total Gap of the requirement appears to be “0”.

5.6.3. Scenario 03

Now, a number of Requirements and Test cases have been taken as Inputs.

Requirements are:

i. Voice Transmission

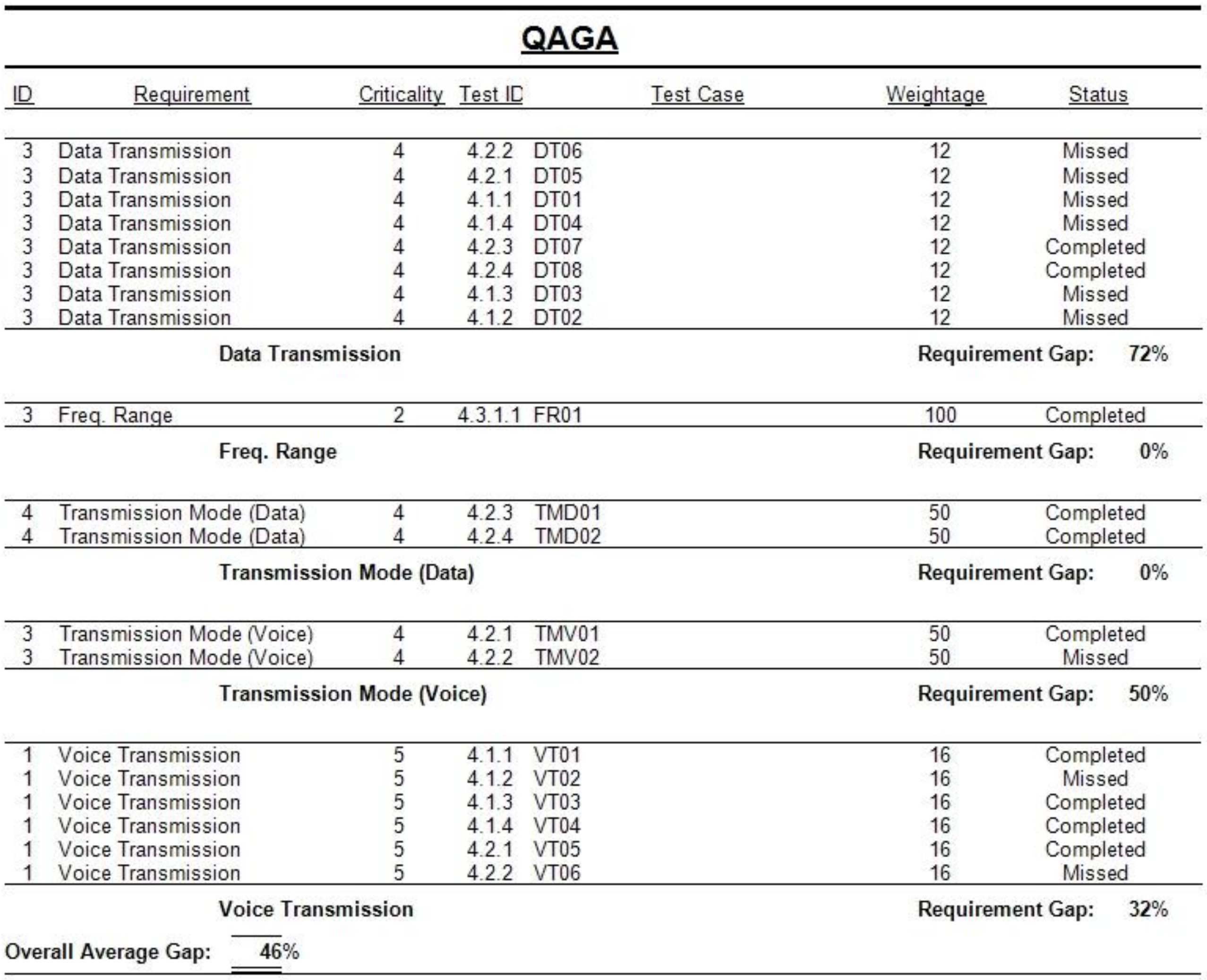
ii. Transmission Mode (Voice)

iii. Transmission Mode (Data)

iv. Frequency Range

v. Data Transmission

The report generated is as follows:



QAGA

ID	Requirement	Criticality	Test ID	Test Case	Weightage	Status
3	Data Transmission	4	4.2.2	DT06	12	Missed
3	Data Transmission	4	4.2.1	DT05	12	Missed
3	Data Transmission	4	4.1.1	DT01	12	Missed
3	Data Transmission	4	4.1.4	DT04	12	Missed
3	Data Transmission	4	4.2.3	DT07	12	Completed
3	Data Transmission	4	4.2.4	DT08	12	Completed
3	Data Transmission	4	4.1.3	DT03	12	Missed
3	Data Transmission	4	4.1.2	DT02	12	Missed
	Data Transmission				**Requirement Gap:**	**72%**
3	Freq. Range	2	4.3.1.1	FR01	100	Completed
	Freq. Range				**Requirement Gap:**	**0%**
4	Transmission Mode (Data)	4	4.2.3	TMD01	50	Completed
4	Transmission Mode (Data)	4	4.2.4	TMD02	50	Completed
	Transmission Mode (Data)				**Requirement Gap:**	**0%**
3	Transmission Mode (Voice)	4	4.2.1	TMV01	50	Completed
3	Transmission Mode (Voice)	4	4.2.2	TMV02	50	Missed
	Transmission Mode (Voice)				**Requirement Gap:**	**50%**
1	Voice Transmission	5	4.1.1	VT01	16	Completed
1	Voice Transmission	5	4.1.2	VT02	16	Missed
1	Voice Transmission	5	4.1.3	VT03	16	Completed
1	Voice Transmission	5	4.1.4	VT04	16	Completed
1	Voice Transmission	5	4.2.1	VT05	16	Completed
1	Voice Transmission	5	4.2.2	VT06	16	Missed
	Voice Transmission				**Requirement Gap:**	**32%**

Overall Average Gap: 46%

Figure 28 The GAP Analysis Report

The graph plotted for the abovementioned Report is as follows:

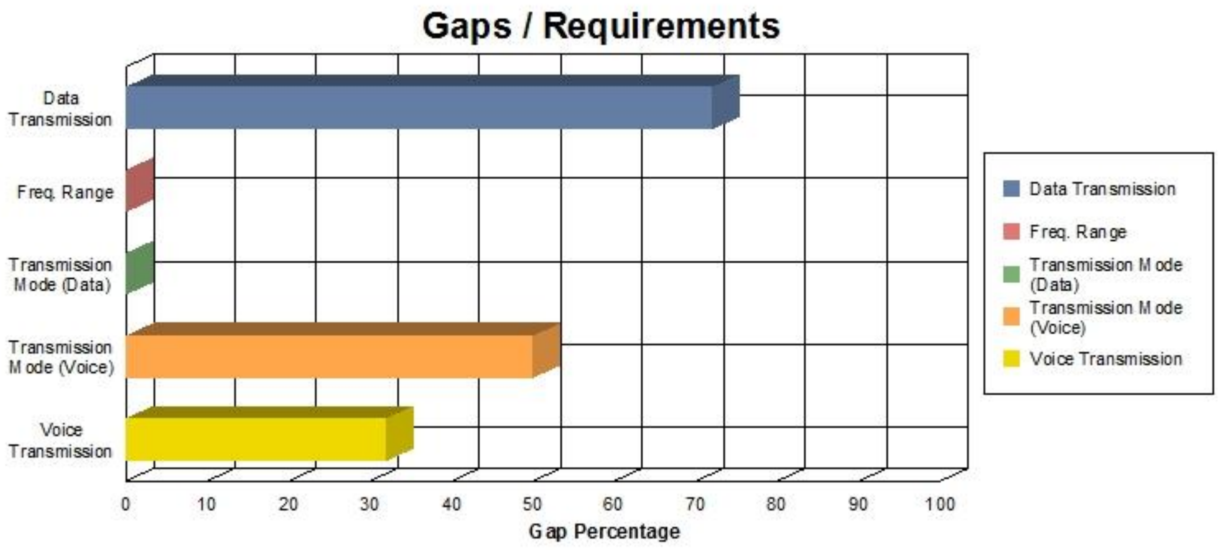


Figure 29 The Gap Analysis Graph

The Report and Graph shows the same results. Requirement “Data Transmission” has a Gap of 72%, “Frequency range” has a Gap of 0%, “Transmission Mode (Data)” has a Gap of 0%, “Transmission Mode (Voice)” has a Gap of 50%, “Voice Transmission” has a Gap of 32%. Thus, the overall Gap of the Software is 46%.

5.6.4. Scenario 04

In another Scenario, Requirements with the name Data Transmission, Voice Transmission, Transmission Mode (Data), Transmission Mode (Voice), Frequency Range and test cases of each requirement is taken as Input with the Criticalities as follows:

Requirement Name	Criticality
Data Transmission	4
Voice Transmission	5
Transmission Mode (Data)	4
Transmission Mode (Voice)	3
Frequency Range	1

Table 4 Requirements and their Criticality

So, the requirement must be plotted in order of the Criticality i.e the requirement with the highest criticality should be plotted on the top and then the other requirements in order of descending criticality.

Hence, the resulting Graph plotted by the tool looks like:

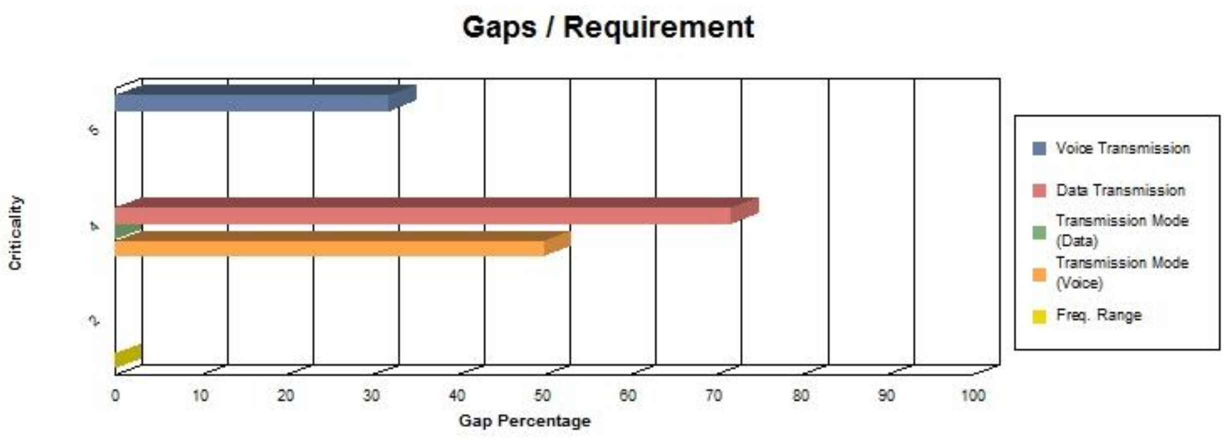


Figure 30 Gap Analysis w.r.t Criticality

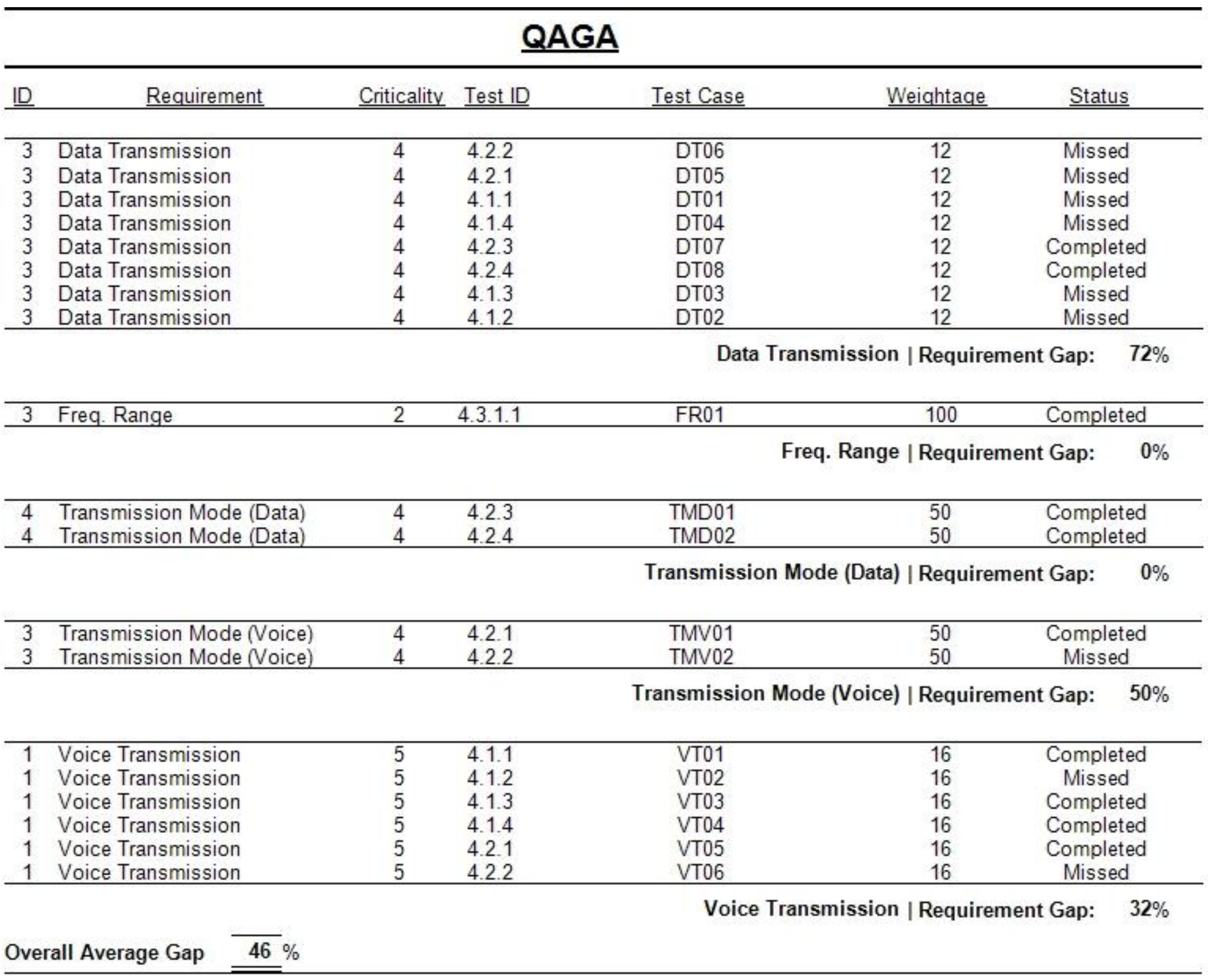
This graph will help Quality Assurance department in knowing the Gap Analysis of the individual requirements with respect to their criticality, so that the requirements with a higher criticality and a larger Gap should be looked after at priority.

As in this particular case, the requirement Voice Transmission has a criticality of 5 and it is plotted on the top, then comes the requirement Data Transmission with criticality 4, and requirement Transmission Mode (Data) has criticality of 4, requirement with the name Transmission Mode (Voice) has a criticality of 3 and finally Frequency Range has a criticality of 1.

So, all these requirements are plotted in the following order which is:

i. Voice Transmission
ii. Data Transmission
iii. Transmission Mode (Data)
iv. Transmission Mode (Voice)
v. Frequency Range

Following is the Report generated by the tool (QAGA) against the same Graph. The Reports depicts the Gap Analysis of each requirement individually including complete detail about every Requirement, Requirement ID, its Criticality and associated Test cases, their Test IDs and Test Weights. Finally an Overall Gap Analysis of the whole Software Project is also presented in the Report.



QAGA

ID	Requirement	Criticality	Test ID	Test Case	Weightage	Status
3	Data Transmission	4	4.2.2	DT06	12	Missed
3	Data Transmission	4	4.2.1	DT05	12	Missed
3	Data Transmission	4	4.1.1	DT01	12	Missed
3	Data Transmission	4	4.1.4	DT04	12	Missed
3	Data Transmission	4	4.2.3	DT07	12	Completed
3	Data Transmission	4	4.2.4	DT08	12	Completed
3	Data Transmission	4	4.1.3	DT03	12	Missed
3	Data Transmission	4	4.1.2	DT02	12	Missed

Data Transmission | Requirement Gap: 72%

3	Freq. Range	2	4.3.1.1	FR01	100	Completed

Freq. Range | Requirement Gap: 0%

4	Transmission Mode (Data)	4	4.2.3	TMD01	50	Completed
4	Transmission Mode (Data)	4	4.2.4	TMD02	50	Completed

Transmission Mode (Data) | Requirement Gap: 0%

3	Transmission Mode (Voice)	4	4.2.1	TMV01	50	Completed
3	Transmission Mode (Voice)	4	4.2.2	TMV02	50	Missed

Transmission Mode (Voice) | Requirement Gap: 50%

1	Voice Transmission	5	4.1.1	VT01	16	Completed
1	Voice Transmission	5	4.1.2	VT02	16	Missed
1	Voice Transmission	5	4.1.3	VT03	16	Completed
1	Voice Transmission	5	4.1.4	VT04	16	Completed
1	Voice Transmission	5	4.2.1	VT05	16	Completed
1	Voice Transmission	5	4.2.2	VT06	16	Missed

Voice Transmission | Requirement Gap: 32%

Overall Average Gap 46 %

Figure 31 Report generated by Tool (QAGA) in order of Criticality of Requirements

5.7. SUMMARY

To fully test the QAGA tool, it has been executed against different Requirement and Test Cases. The reports of the Gap Analysis of the individual requirements as well as for the whole Software projects have been generated along with the charts plotted for every requirement. Extensive testing shows that the proposed Model and its implementation can play an effective role in helping organization to know the status of the Quality Assurance of their products.

Chapter 6

CONLUSIONS AND FUTURE WORK

All the software companies in Pakistan are still not CMMI certified which is being a great hurdle for in and off shore clients' attraction. Developing a Model which can assist software companies in developing their projects according to the SEI standards can make our software industry far more progressive and advanced.

6.1. CONCLUSIONS

Checking the compliance of a software project for a predefined project-specific process is helpful to quality assurance. In addition, compliance checking of actual performed processes is also supportive in process improvement.

In this thesis an attempt has been made to propose a new Model which helps organization in attaining CMMI level 2 Process and Product Quality Assurance.

The primary contributions of the research effort are:

A generic model and a tool:

i. That would provide the organization with a guideline to achieve the organizational quality assurance objectives.

ii. That will present the status of the chosen project/process after having the execution of tool.

iii. To provide the stakeholders with necessary information and basis to make informed decision afterwards in order to improve the chosen processes/projects.

iv. Helping the organization in achieving next CMMI level.

6.2. FUTURE WORK

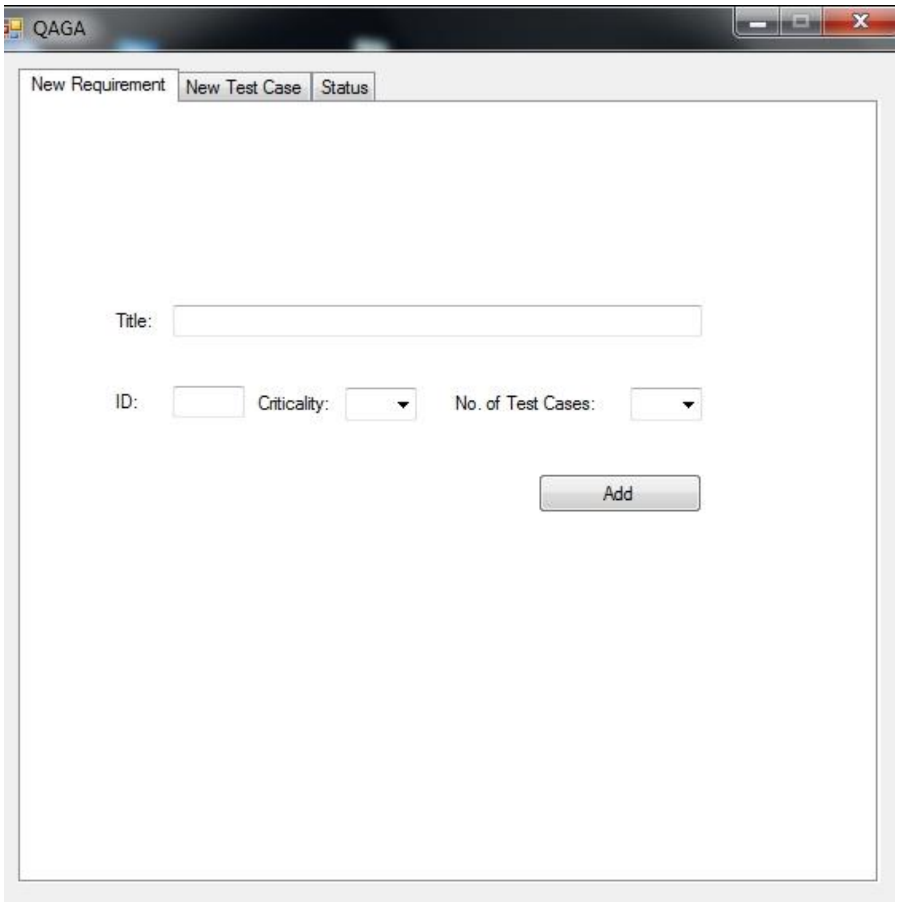
The proposed model when implemented showed fruitful result in measuring the Gaps in the Quality of a Software Project but there is still room for improvement. Future work can be carried out in designing a model for Automatic Quality Assurance Gap Analysis in CMMI Level 2 for Compliance. The tool presented can be tweaked for reading the inputs from the file automatically and then generate the results.

Another possibility is the proposal of more efficient and effective models to cover the Quality Assurance Process Area of CMMI.

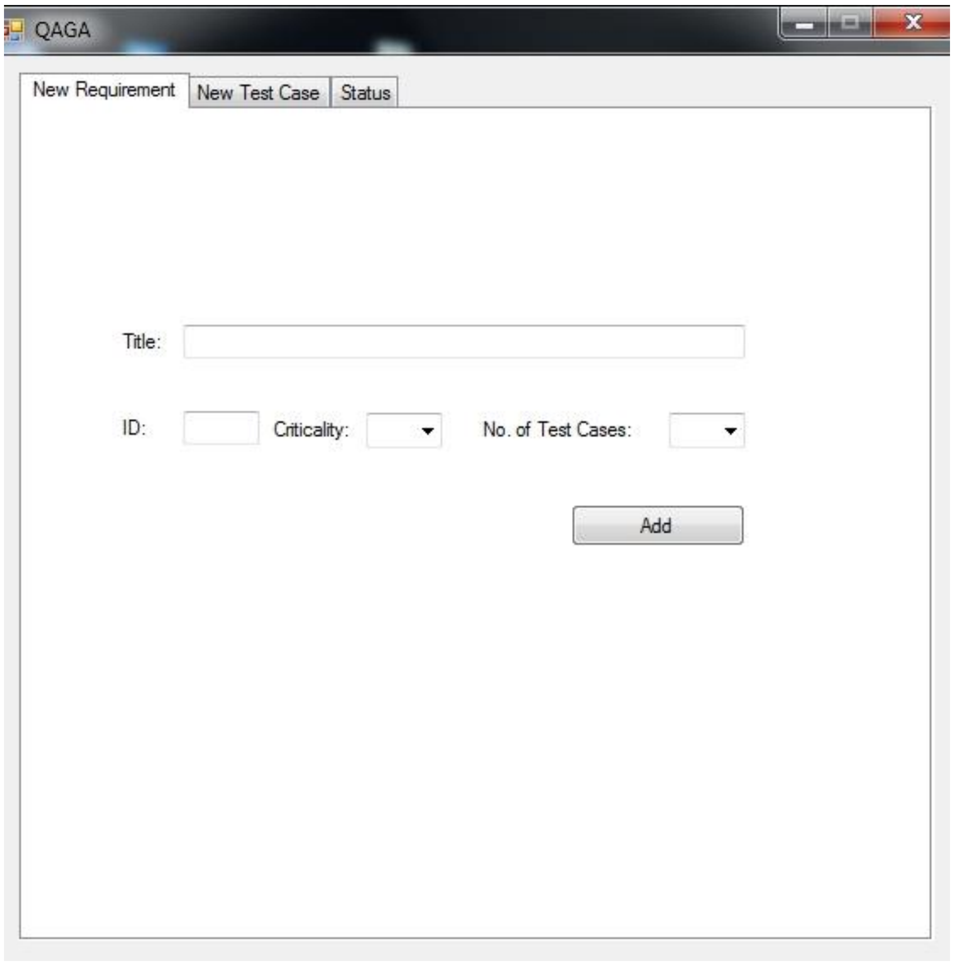
Appendix A

SNAPSHOTS

A.1 GUI of QAGA



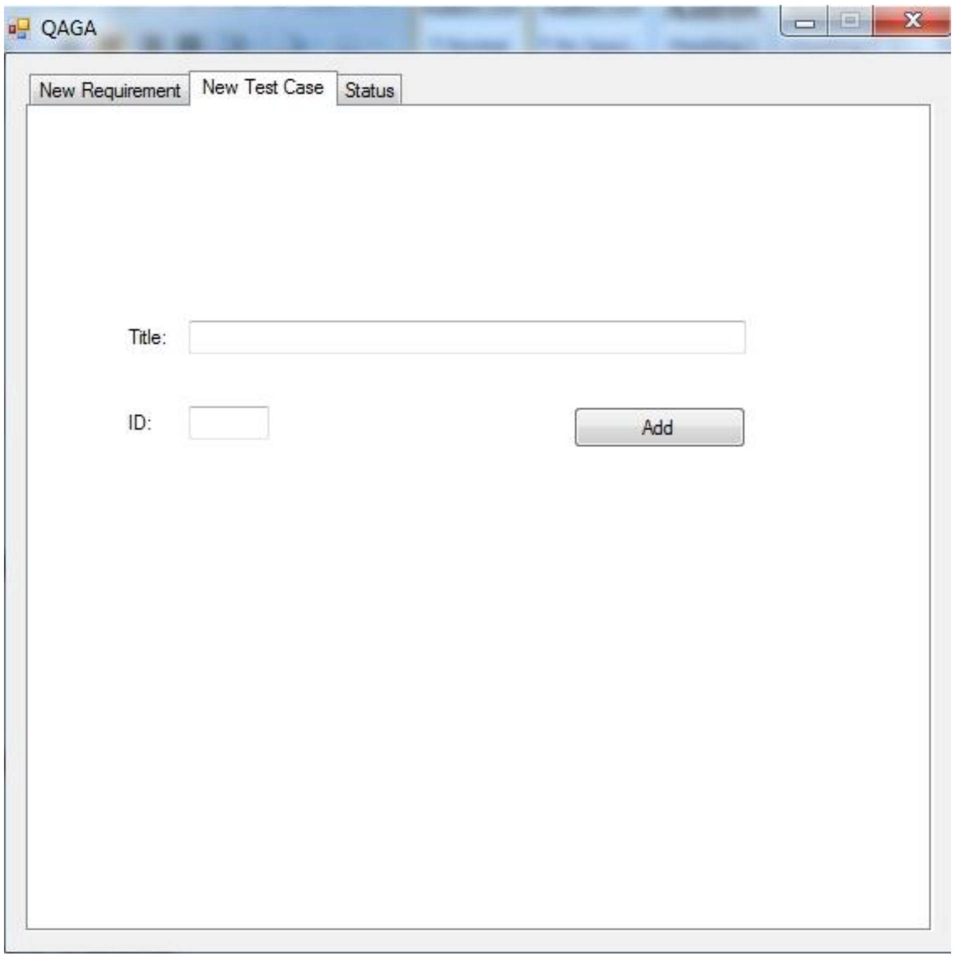
A.2 GUI of Requirement Input



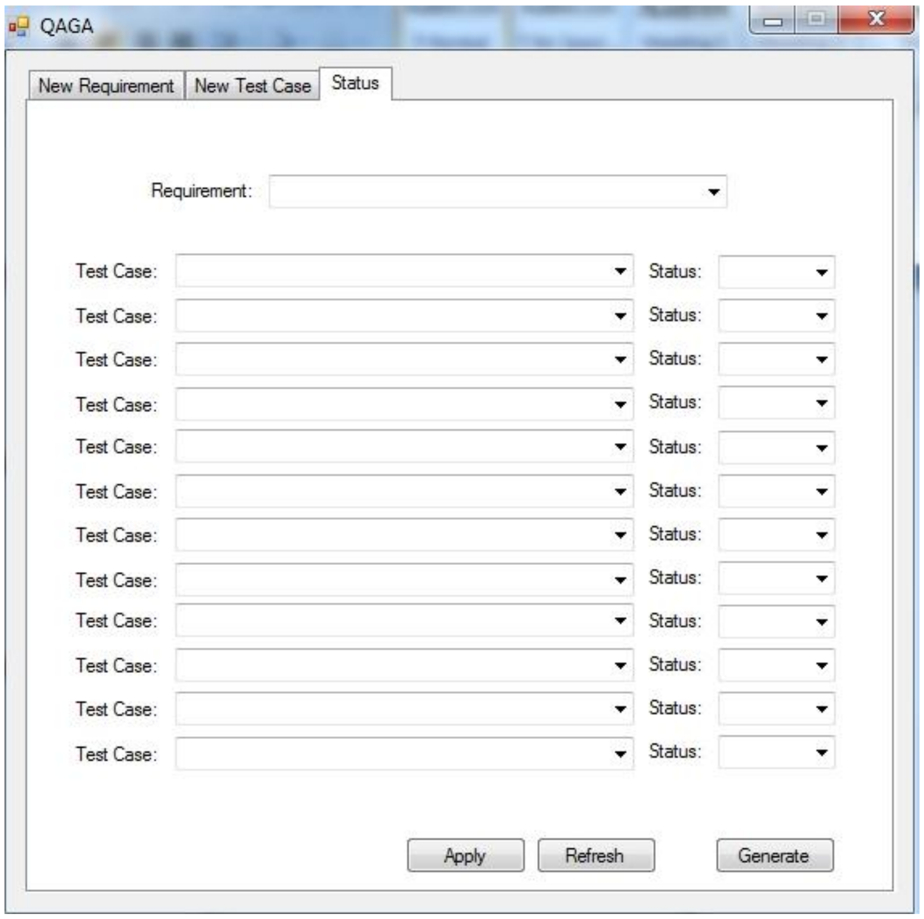
Title:

ID: Criticality: No. of Test Cases:

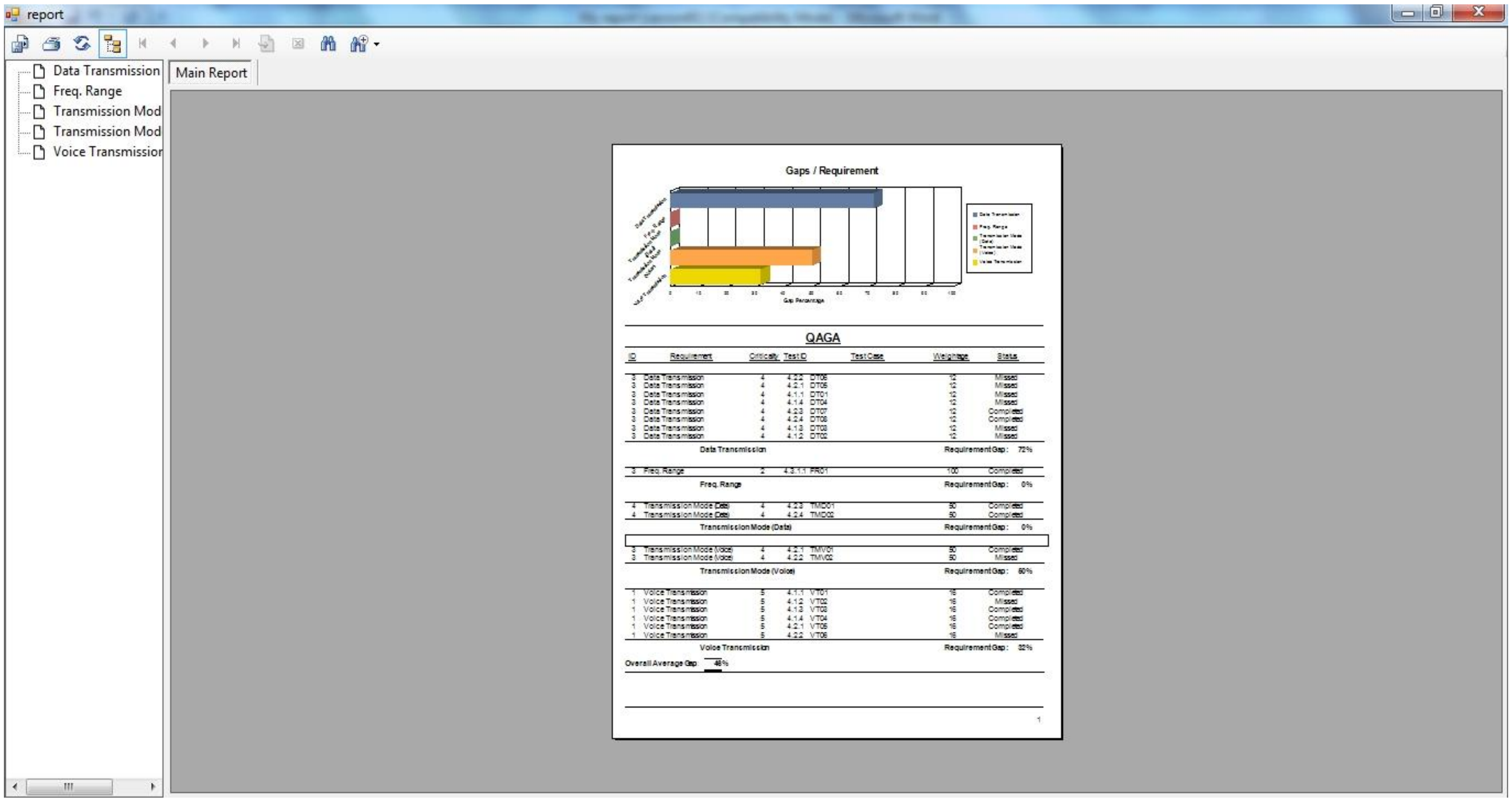
A.3 GUI of Test Case Input



A.4 GUI of Mapping between Requirements and Test Cases



A.5 GUI of Report



QAGA

ID	Requirement	Criticaly	Test ID		Test Case	Weightage	Status
3	Data Transmission	4	4.2.2	DT06		12	Missed
3	Data Transmission	4	4.2.1	DT05		12	Missed
3	Data Transmission	4	4.1.1	DT01		12	Missed
3	Data Transmission	4	4.1.4	DT04		12	Missed
3	Data Transmission	4	4.2.3	DT07		12	Completed
3	Data Transmission	4	4.2.4	DT08		12	Completed
3	Data Transmission	4	4.1.3	DT03		12	Missed
3	Data Transmission	4	4.1.2	DT02		12	Missed
	Data Transmission					Requirement Gap :	72%
3	Freq. Range	2	4.3.1.1	FR01		100	Completed
	Freq. Range					Requirement Gap :	0%
4	Transmission Mode (Data)	4	4.2.3	TMD01		50	Completed
4	Transmission Mode (Data)	4	4.2.4	TMD02		50	Completed
	Transmission Mode (Data)					Requirement Gap :	0%
3	Transmission Mode (Voice)	4	4.2.1	TMV01		50	Completed
3	Transmission Mode (Voice)	4	4.2.2	TMV02		50	Missed
	Transmission Mode (Voice)					Requirement Gap :	50%
1	Voice Transmission	5	4.1.1	VT01		16	Completed
1	Voice Transmission	5	4.1.2	VT02		16	Missed
1	Voice Transmission	5	4.1.3	VT03		16	Completed
1	Voice Transmission	5	4.1.4	VT04		16	Completed
1	Voice Transmission	5	4.2.1	VT05		16	Completed
1	Voice Transmission	5	4.2.2	VT06		16	Missed
	Voice Transmission					Requirement Gap :	32%

Overall Average Gap: 46%

1

REFERENCES

[1] ANSI/ASQC A3-1978.

[2] Pressman, Scott (2005), Software Engineering: A Practitioner's Approach (Sixth, International ed.), McGraw-Hill Education Pressman, p. 388.

[3] ANSI/IEEE Std. 730-1981, IEEE Standard for Software Quality Assurance Plans, 345 East 47th Street, New York, NY 10017, Nov.13, 1981

[4] http://asq.org/learn-about-quality/quality-assurance-quality control/overview/overview.html

[5] ANSI N 45.2.10-1973.

[6] R. Fairley, Software Quality Engineering Course Notes. 16. April 1997.

[7] ISO/IEC 12207, Systems and Software Engineering – Software life cycle processes. ISO, Geneva, 2007, p.6.

[8] Nielsen, David, “CMM and Project Quality Management”.

[9] Jeffrey Voas, Cigital, "Assuring Software Quality Assurance"

[10] Raymond J. Rubey Audrey C. Brewer, Software Quality Assurance Standards -- A Comparison and an Integration] The first SQA standard to be widely distributed and used was MIL-S-52779 [MIL-S-52779A. Military Spec@carion, Sofnvare Quality Assurance Program Requirements, 1 August 1979.

[11] MIL-S-52779A. Military Spec@carion, Sofnvare Quality Assurance Program Requirements, 1 August 1979.

[12] http://www.sei.cmu.edu/cmmi/

[13] http://www.tutorialspoint.com/cmmi/cmmi-process-areas.htm.

[14] Raymond J. Rubey, Audrey C. Brewer, "Software Quality Assurance Standards -- A Comparison and Integration".

[15] Charles H. Wells, Russell Brand and Lawrence Markosian, "Customized Tools for Software Quality Assurance and Reengineering".

[16] Miguel A. Serrano, Carlos Montes de Oca, Karina Cedillo, "An Experience on using the Team Software Process for Implementing the Capability Maturity Model for Software in a Small Organization”.

[17] Luis Manuel Gonzalez Amaral, João Pascoal Faria, "A Gap Analysis Methodology for the Team Software Process".

[18] Lars M. Karg, Michael Grottke, Arne Beckhaus, "Conformance Quality and Failure Costs in the Software Industry: An Empirical Analysis of Open Source Software".

[19] Tobias Otte, Robert Moreton, Heinz D. Knoell, "Development of a Quality Assurance Framework for the Open Source Development Model".

[20] Yas A. Alsultanny, Ahmed M. Wohaishi, "Requirements of Software Quality Assurance Model".

[21] Mahmood Niazi, Muhammad Ali Babar, "Identifying high perceived value practices of CMMI level 2: An empirical study"

[22] Roger S. Pressman, “Software Engineer – A practitioner’s approach”.

[23] CMMI-ACQ (Version 1.3, November 2010). Carnegie Mellon University Software Engineering Institute. 2010. Retrieved 16 February 2011.

[24] "Appraisal Requirements for CMMI, Version 1.2 (ARC, V1.2)" Carnegie Mellon University Software Engineering Institute. 2006. Retrieved 16 February 2011.

[25] http://www.benlinders.com/2011/cmmi-v1-3-summing-up/

[26] http://www.benlinders.com/2010/cmmi-v1-3-agile/

[27] http://www.benlinders.com/2010/cmmi-v1-3-released-high-maturity-clarified

[28] http://www.benlinders.com/2010/cmmi-v1-3-deploying-the-cmmi

[29] http://www.tutorialspoint.com/cmmi/cmmi-process-areas.htm