1

INTRODUCTION TO QUALITY

OBJECTIVES

This chapter provides a basic understanding of quality management. It describes basic definitions of quality and premises of quality management with different views of product quality from stakeholder's perspectives. This foundation is essential for software testers to understand the criticality of their position in software development life cycle.

1.1 INTRODUCTION

Evolution of mankind can be seen as a continuous effort to make thing better and convenient through improvements linked with inventions of new products. Since time immemorial, humans have used various products to enhance their lifestyle. Initially, mankind was completely dependent on nature for satisfying their basic needs of food, shelter and clothing. With evolution, human needs increased from the basic level to substantial addition of derived needs to make life more comfortable. As civilisation progressed, humans started converting natural resources into things which could be used easily to satisfy their basic as well as derived needs.

Earlier, product quality was governed by the individual skill and it differed from instance to instance depending upon the creator and the process used to make it at that instance. Every product was considered as a separate project and every instance of the manufacturing process led to products of different quality attributes. Due to increase in demand for same or similar products which were expected to satisfy same/similar demands and mechanisation of the manufacturing processes, the concept of product specialisation and mass production came into existence. Technological developments made production faster and repetitive in nature.

Now, we are into the era of specialised mass production where producers have specialised domain knowledge and manufacturing skill required for particular product creation, and use this knowledge and skill to produce the product in huge quantities. The market is changing considerably from monopoly to fierce competition but still maintaining the individual product identity in terms of attributes and characteristics. There are large number of buyers as well as sellers in the market, providing and demanding similar products, which satisfy similar demands. Products may or may not be exactly the same but may be similar or satisfying similar needs/demands of the users. They may differ from one another to some extent on the basis of their cost, delivery schedule to acquire it, features, functionalities present/absent, etc. These may be termed as attributes of the quality of a product.

We will be using the word 'product' to represent 'product' as well as 'service', as both are intended to satisfy needs of users/customers. Service may be considered as a virtual product without physical existence but satisfying some needs of users.

1.2 HISTORICAL PERSPECTIVE OF QUALITY

Quality improvement is not a new pursuit for mankind. The field of quality and quality improvement has its roots in agriculture. Early efforts of quality improvement in agriculture may be attributed to statistical research conducted in Britain, in early 20th century, to assist farmers in understanding how to plan the crops and rotate the plan of cultivation to maximise agricultural production while maintaining the soil quality at the same time.

This work inspired Walter Shewhart at Bell Laboratories to develop quality improvement programs through planned efforts. He adopted the concepts developed initially for agriculture to implement quality improvement programs for products and to reduce the cost to customer without affecting profitability for the manufacturer. Changes brought in by Walter Shelwart motivated Dr Edward Deming to implement quality improvement programs as a way to improve product quality. He devoted his life to teaching and improving quality methods and practices across the world through 'Total Quality Management' methodology.

Dr Deming demonstrated his ideas of 'Total Quality Management' through continual improvement in Japan. Dr Joseph Juran also implemented quality improvement through measurement programs using different quality tools for assessment and improvement. Japanese producers fully embraced quality improvement methodologies and started to integrate the concepts of 'Total Quality Management' in their industries. The Dramatic improvement in quality of products in Japan after 1950 due to the revolutionary ideas of continual improvement through process measurement are still considered legendary.

During last few decades, the Japanese industry has successfully utilised quality tools and 'Total Quality Management' methodologies as part of their successful effort to become a leading nation in manufacturing and supplying a vast array of electronics, automotive and other products to the entire world. Quality of the products is established and continually improved in terms of features, consistent performance, lesser costs, reasonable delivery schedule, etc. in order to enhance the satisfaction of the customer. Japanese products started dictating the quality parameters in world market to the extent that many nations adopted quality improvement programs at national level to face the competition from the Japanese industry. Quality of Japanese products stems from the systematic organisation and understanding of processes used in all aspects of product development, and introduction of tools and methodologies that permit monitoring and understanding about what is happening in different processes of manufacturing and management of interactions of those processes. Japanese quality improvement programs created the sets of interrelated processes which assure the same product quality in repetitive manner and in large number to satisfy the demand of a huge market. Defects are analysed and root causes of the defects are identified and eliminated through continual process improvement. This has helped in optimising the processes to produce better results in repetitive manner.

1.3 WHAT IS QUALITY? (IS IT A FACT OR PERCEPTION?)

What is quality, is an important question but does not have a simple answer. Some people define it as a fact while others define it as a perception of customer/user.

We often talk about quality of a product to shortlist or select the best product among the equals when we wish to acquire one. We may not have a complete idea about the meaning of quality or what we are looking for while selecting a product, if somebody questions us about the reason for choosing one product over the

other. This is a major issue faced by the people working in quality field even today as it is very difficult to decide what contributes customer loyalty or first-time sale and subsequent repeat sale. The term 'quality' means different things to different people at different times, different places and for different products. For example, to some users, a quality product may be one, which has no/less defects and works exactly as expected and matches with his/her concept of cost and delivery schedule along with services offered. Such a thought may be a definition of quality—**'Quality is fitness for use'**.

However, some other definitions of quality are also widely discussed. Quality defined as, **'Conformance to specifications'** is a position that people in the engineering industry often promote because they can do very little to change the design of a product and have to make a product as per the design which will best suite the user's other expectations like less cost, fast delivery and good service support. Others promote wider views, which may include the attribute of a product which satisfies/exceeds the expectations of the customer. Some believe that quality is a judgment or perception of the customer/user about the attributes of a product, as all the features may not be known or used during the entire life of a product. Quality is the extent to which the customers/users believe that the product meets or surpasses their needs and expectations. Others believe that quality means delivering products that,

- Meet customer standards, either as defined by the customer or defined by the normal usage or by some national or international bodies. (Standard may or may not be as defined by the supplier of the product. Typically for consumer goods, standard is defined by market forces and likes and dislikes of users in general.)
- Meet and fulfill customer needs which include expressed needs as well as implied requirements derived by business analysts and system analysts. Expressed needs are available in the form of requirement statement generated by users while implied needs definition may require supplier to understand customer business and provide the solution accordingly.
- One must try to meet customer expectations as maximum as possible. If something is given more than the requirements of the customer, it should be declared before transition, so that customer surprises can be avoided. At the same time, if some aspect of the specified requirements has not been included in the product, it should be declared. This is essential so that the customer may understand the deliverables accordingly. Expectations may be at the top of customer needs and may be useful in creating brand loyalty through customer delight. (Trying to get customer delight after informing customer about it.)
- Meet anticipated/unanticipated future needs and aspirations of customers by understanding their businesses and future plans. One may need to build a software and system considering some future requirements. Every product including software has a life span and due to technological inventions as well as new ways of doing things, older systems become obsolete, either technically or economically. How much of the future must be considered for the given product may be a responsibility of the customer or the supplier or a joint responsibility. Every product has some defined life span and one may have to extrapolate future needs accordingly.

Others may simply ignore these definitions of quality and say, **'I'll know the quality of a product when I see it'**. It seems that we all **'know'** or **'feel'** somehow what the meaning of quality is, though it is very difficult to put it in exact words. Something that fulfills/exceeds customer's preconceived ideas about the quality is likely to be called as a quality product.

We will try to examine tools and methods which can be used to improve product quality through process approach, add value through brainstorming by producers, consumers, customers and all stakeholders about new features which may be included/old features which may be excluded from the product, decrease costs,

improve schedule and help products to conform better with respect to the expressed and implied requirements. Use of quality tools and methodologies can help people engage in production related activities to improve quality of the products delivered to final user and achieve customer satisfaction.

1.4 DEFINITIONS OF QUALITY

For achieving quality of a product, one must define it in some measurable terms which can be used as a reference to find whether quality is really met or not. There are many views and definitions of quality given by stalwarts working in quality improvement and quality management arena. These definitions describe different perceptions toward quality of products. Some of these are,

1. Customer-Based Definition of Quality Quality product must have **'Fitness for use'** and must meet customer needs, expectations and help in achieving customer satisfaction and possibly customer delight. Any product can be considered as a quality product if it satisfies its purpose of existence through customer satisfaction. This definition is mainly derived by an approach to quality management through **'Quality is fitness for use'.**

2. Manufacturing-Based Definition of Quality This definition is mainly derived from engineering product manufacturing where it is not expected that the customer knows all requirements of the product, and many product level requirements are defined by architects and designers on the basis of customer feedback/survey. Market research may have to generate requirement statement on the basis of perception of probable customers about what features and characteristics of a product are expected by the market. A quality product must have a definition of requirement specifications, design specifications, etc. and the product must conform to these specifications. The development methodologies used for the purpose must be capable of producing the right product in first go and must result into a product having no/minimum defects. This approach gives the definition of **'Conformance to requirements'.**

3. Product-Based Definition of Quality The product must have something that other similar products do not have which can help the customer satisfy his/her needs in a better way. These attributes must add value for the customer/user so that he/she can appreciate the product in comparison to competing products. This makes the product distinguishable from similar products in the market. Also, the customers must feel proud of owning it due to its inherent attributes and characteristics.

4. Value- Based Definition of Quality A product is the best combination of price and features or attributes expected by or required by the customers. The customer must get value for his investment by buying the product. The cost of a product has direct relationship with the value that the customer finds in it. More value for the customer helps in better appreciation of a product. Many times it is claimed that **'People do not buy products, they buy benefits'.**

5. Transcendent Quality To many users/customers, it is not clear what is meant by a 'quality product', but as per their perception it is something good and they may want to purchase it because of some quality present/absent in the product. The customer will derive the value and may feel the pride of ownership. Definitions 2, 3 and 4 are traditionally associated with the idea of a product quality that,

- A product must have zero/minimum defects so that it does not prohibit normal usage by the users. When users buy a product, they expect minimum/no failures.

- A product must be something that people will want to receive as it satisfies their needs and supports their expectations. It must suffice the purpose of its existence from the customers view.
- It can be purchased at a reasonable price with relation to the value users may derive from it. The customer may undertake cost benefit analysis of the product and if benefits equal or exceed cost, it may be bought.

Customer centric product development forces the industry to look outside its own premises and thought process, making it compulsory to understand the customer. This forces the producer to create products that prospective customers may want to buy and not ones that designers think people want to receive.

The most interesting definition of quality is, **'I do not know what it is, but if I'm delighted by acquiring it, I'll buy it!'** This means that those products are better in quality which possess some characteristics that attract customers to purchase them. Though the requirements of a product may differ from customer to customer, place to place and time to time, in general, the product must be,

- Less expensive, with higher returns or higher values for the customer, satisfying cost benefit analysis. Directly or indirectly, every owner would be performing cost benefit analysis before arriving at a decision to purchase something.
- Inherent of required features or attributes expected by the customer which make it fit for use. If product is of no use to users, they will never purchase it.
- Without any defect or with few defects so that its usage is uninhibited and failure or repairs would be as less as possible. If the product is very reliable, it may be liked by prospective buyers.
- With desirable cosmetic attributes.

Often, we are not aware that we want a certain product, but when we see their attributes, we feel like buying it. It may include cosmetic requirements like user interfaces, ease of use, etc.

Many of the above statements are based upon the users or customers perception about quality of the product that they wish to buy. Some of these characteristics are often attributes of products delivered by Japanese manufacturers to consumer market. Mainly, the contributors to quality would be that the failure rates are less, repairs are easier and fast, products are consistent in performance and work better than other similar products in the market and do not give any surprises to customers during use. The reasons for such improvement in the quality are a continuous/continual improvement in all aspects of product development through requirement capturing, design, development, testing, deployment and maintenance.

There is one more angle to the definition of quality of a product. Any improvement in the product quality must result into a better product, and it must give benefits to the customer finally. The benefits may be in terms of more or better features, less wait time, less cost, better service, etc. Thus, quality improvement has direct relationship with fulfilling customer requirements or giving more and more customer satisfaction.

Many people speak about not only achieving customer satisfaction but exceeding customer expectations to achieve customer delight. There is a signal of caution about exceeding customer expectations. Any feature which surprises a customer may not be appreciated by him/her and may be termed as a defect. Exceeding expectations without informing the customer about what they can expect in addition to defined requirements can be dangerous and may result in rejection of such products.

1.5 CORE COMPONENTS OF QUALITY

Quality of a product must be driven by customer requirements and expectations from the product. Those expectations may be expressed as a part of requirement specifications defined or may be implied one which is generally accepted as requirements. It must have some important characteristics that may help

customer in getting more and more benefits and satisfaction by using the product. Some postulates of quality are,

1.5.1 QUALITY IS BASED ON CUSTOMER SATISFACTION BY ACQUIRING A PRODUCT

Quality is something perceived by a customer while using a product. The effect of a quality product, delivered and used by a customer, on his satisfaction and delight is the most important factor in determining whether the quality has been achieved or not. It talks about the ability of a product or service to satisfy a customer by fulfilling his needs or purpose for which it is acquired. It may come through the attributes of a product, time required for a customer to acquire it, price a customer is expected to pay for it and so many other factors associated with the product as well as the organisation producing or distributing it. All these factors may or may not be governed by the manufacturer alone but may be dependent on quality of inputs. This point stresses a need that a producer must understand the purpose or usage of a product and then devise a quality plan for it accordingly, to satisfy the purpose of the product.

1.5.2 THE ORGANISATION MUST DEFINE QUALITY PARAMETERS BEFORE IT CAN BE ACHIEVED

We have already discussed that quality is a perception of a customer about satisfaction of needs or expectation. It is difficult for the manufacturer to achieve the quality of product without knowing what customer is looking for while purchasing it. If product quality is defined in some measurable terms, it can help the manufacturer in deciding whether the product quality has been achieved or not during its manufacturing and delivery. In order to meet some criteria of improvement and ability to satisfy a customer, one must follow a cycle of 'Define', 'Measure', 'Monitor', 'Control' and 'Improve'. The cycle of improvements through measurements is described below,

- *Define* There must be some definition of what is required in the product, in terms of attributes or characteristics of a product, and in how much quantity it is required to derive customer satisfaction. Features, functionalities, and attributes of the product must be measured in quantitative terms, and it must be a part of requirement specification as well as acceptance criteria defined for it. The supplier as well as the customer must know what 'Must be', what 'Should be' and what 'Could be' present in the product so delivered and also what 'Must not be', 'Should not be' and 'Could not be' present in the product.

- *Measure* The quantitative measures must be defined as an attribute of quality of a product. Presence or absence of these attributes in required quantities acts as an indicator of product quality achievement. Measurement also gives a gap between what is expected by a customer and what is delivered to him when the product is sold. This gap may be considered as a lack of quality for that product. This may cause customer dissatisfaction or rejection by the customer.

- *Monitor* Ability of the product to satisfy customer expectations defines the quality of a product. There must be some mechanism available with the manufacturer to monitor the processes used in development, testing and delivering a product to a customer and their outcome, i.e. attributes of product produced using these processes, to ensure that customer satisfaction is incorporated in the deliverables given to the customer. Deviations from the specifications must be analysed and reasons of these deviations must be sorted out to improve product and process used for producing it. An organisation must have correction as well as corrective and preventive action plans to remove the reasons of deviations/deficiencies in the product as well as improve the processes used for making it.

• *Control* Control gives the ability to provide desired results and avoid the undesired things going to a customer. Controlling function in the organisation, popularly called as 'quality control' or 'verification and validation', may be given a responsibility to control product quality at micro level while the final responsibility of overall organisational control is entrusted with the management, popularly called as 'quality assurance'. Management must put some mechanism in place for reviewing and controlling the progress of product development and testing, initiating actions on deviations/deficiencies observed in the product as well as the process.

• *Improve* Continuous/continual improvements are necessary to maintain ongoing customer satisfaction and overcome the possible competition, customer complaints, etc. If some producer enjoys very high customer satisfaction and huge demand for his product, competitors will try to enter the market. They will try to improve their products further to beat the competition. Improvement may be either of two different approaches viz. continuous improvement and continual improvement as the case may be.

1.5.3 MANAGEMENT MUST LEAD THE ORGANISATION THROUGH IMPROVEMENT EFFORTS

Quality must be perceived by a customer to realise customer satisfaction. Many factors must be controlled by a manufacturer in order to attain customer satisfaction. Management is the single strongest force existing in an organisation to make the changes as expected by a customer; it naturally becomes the leader in achieving customer satisfaction, quality of product and improvement of the processes used through various programs of continuous/continual improvement. Quality management must be driven by the management and participated by all employees.

Management should lead the endeavor of quality improvement program in the organisation by defining vision, mission, policies, objectives, strategies, goals and values for the organisation and show the existence of the same in the organisation by self examples. Entire organisation should imitate the behavioral and leadership aspects of the management. Every word and action by the management may be seen and adopted by the employees. Quality improvement is also termed as a 'cultural change brought in by management'.

Organisation based policies, procedures, methods, standards, systems etc. are defined and approved by the management. Adherence to these systems must be monitored continuously and deviation/deficiencies must be tracked. Actions resulting from the observed deviations/deficiencies shall be viewed as the areas which need improvements. The improvements may be required in enforcement or definition of policies and procedures, methods, standards, etc. Management must have quality planning at organisation level to support improvement actions.

1.5.4 CONTINUOUS PROCESS (CONTINUAL) IMPROVEMENT IS NECESSARY

There was an old belief that quality can be improved by more inspection, testing and rework, scrap, sorting, etc. It was expected that a customer must inspect the product and report the defects or deficiencies so that those will be fixed. Manufacturer was responsible for fixing the defects as and when they were reported by the customer. This added to the cost of inspection, segregation, failure, rework, etc. for the customer and reduced the profit margins for the manufacturer or increased the price for the customer. At the same time, customer's right to receive a good product was withdrawn.

For improving the competitive cost advantage to producer as well as customer, quality must be produced with an aim of first time right and must be improved continuously/continually. For the customer, total cost of product is inclusive of cost of purchase and maintenance. Total cost is more important to him than the purchase tag. The first step for producing quality is the definition of processes used for producing the product and the cycle of

continuous or continual improvement (Plan–Do–Check–Act or Define–Measure–Monitor–Control–Improve) to refine and redefine processes to achieve targeted improvements. It needs 'Planning' for quality, 'Doing' as per the defined plans, 'Checking' the outcome at each stage with the expected results and taking 'Actions' on the variances produced. Refer Table 1.1 for comparison between continuous and continual improvement.

Table 1.1 Comparison of continuous and continual improvement

Continuous improvement	Continual improvement
Continuous improvement is dynamic in nature. The changes are done at every stage and every time to improve further.	Continual improvement is dynamic as well as static change management. The changes are done, absorbed, baselined and sustained before taking next step of improvements.
Continuously striving for excellence gives a continuous improvement	Periodic improvements followed by stabilisation process and sustenance represent continual improvement
It has a thrust on continuous refinement of the processes to eliminate waste continuously	Stabilisation of processes at each iteration of improvement where waste is removed in stages
It has high dependence on people having innovative skills tending towards inventions	Less dependence on people and more dependence on innovation processes
Environment is continuously changed	Changes in environment are followed by stabilisation
Sometimes it creates a turbulence in an organisation, if people are not able to digest continuous change	It may be better suited than continuous improvement. It gives a chance to settle the change before next change is introduced

1.6 QUALITY VIEW

Stakeholders are the people or entities interested in success/failure of a project or product or organisation in general. Every project/product/organisation has several stakeholders interested in its betterment. Quality is viewed differently by different stakeholders of the product/project/organisation as per their role in entire spectrum. Some quality models put all stakeholders for the project and product in six major categories. These stakeholders benefit directly or indirectly, if the project/product/organisation becomes successful, and suffer, if the organisation or project fails.

- **Customer** Customer is the main stakeholder for any product/project. The customer will be paying for the product to satisfy his requirements. He/she must benefit by acquiring a new product. Sometimes, the customer and user can be different entities but here, we are defining both as same entity considering customer as a user. Though sometimes late delivery penalty clauses are included in contract, the customer is interested in the product delivery with all features on defined scheduled time and may not be interested in getting compensated for the failures or delayed deliveries.
- **Supplier** Suppliers give inputs for making a project/product. As an organisation becomes successful, more and more projects are executed, and suppliers can make more business, profit and expansion. Suppliers can be external or internal to the organisation. External suppliers may include people supplying machines, hardware, software, etc. for money, while internal suppliers may include other functions such as system administrator, training provider, etc. which are supporting projects/product development.

• **Employee** People working in a project/an organisation may be termed as employees. These people may be permanent/temporary workers but may not be contractual labours having no stake in product success. (Contractual workers may come under supplier category.) As the projects/organisations become successful, people working on these projects/in these organisations get more recognition, satisfaction, pride, etc. They feel proud to be part of a successful mission.

• **Management** People managing the organisation/project may be termed as management in general. Management may be divided further into project management, staff management, senior management, investors, etc. Management needs more profit, recognition, turnover improvements, etc to make their vision and mission successful. Successful projects give management many benefits like expanding customer base, getting recognition, more profit, more business, etc.

There are two more stakeholders in the success as well as failure of any project/product/organisation. Many times, we do not feel their existence at project level or even at organisation level. But they do exist at macro level.

• **Society** Society benefits as well as suffers due to successful projects/organisations. It is more of a perception of an individual looking towards the success of the organisation. Successful organisations/ projects generate more employment, and wealth for the people who are in the category of customer, supplier, employee, management, etc. It also affects the resource availability at local as well as global level like water, roads, power supply, etc. It also affects economics of a society to a larger extent. Major price rise has been seen in industry dominated areas as the paying capacity of people in these areas is higher than other areas where there is no such industry.

• **Government** Government may be further categorised as local government, state government, central government, etc. Government benefits as well as suffers due to successful projects/organisations. Government may get higher taxes, export benefits, foreign currency, etc. from successful projects/organisations. People living in those areas may get employment and overall wealth of the nation improves. At the same time, there may be pressure on resources like water, power, etc. There may be some problems in terms of money availability and flow as success leads to more buying power and inflation.

Quality perspective of all these stakeholders defines their expectations from organisation/projects. We may feel that superficially these views may differ from each other though finally they may be leading to the same outcome. If these views match perfectly and there is no gap in the stakeholder's expectations, then organisational performance and effectiveness can be improved significantly as collective efforts from all stakeholders. If the views differ significantly, this may lead to discord and hamper improvement.

Let us discuss two important views of quality which mainly defines the expectations from a project and success of a project at unit level viz. customer's view and developer's view of quality.

1.6.1 CUSTOMER'S VIEW OF QUALITY

Customer's view of quality of product interprets customer requirements and expectation for getting a better product at defined schedule, cost and with adequate service along with required features and functionalities. Customer is paying some cost to get a product because he finds value in such acquisition.

Delivering Right Product The products received by customers must be useful to satisfy their needs and expectations. It may or may not be the correct product from manufacturer's perspective or what business analyst/system designer may think. There is a possibility that development team including testers

may ask several queries about the requirements of the product to get them clarified but the final decision about the requirements definition shall be with customer. If customer confirms that the requirements are correct/not correct, then there is no possibility of further argument about the validity/invalidity of requirements.

Satisfying Customer's Needs The product may or may not be the best product, as per the manufacturer's views or those which are available in the market, which can be made from the given set of requirements and constraints. There are possibilities of different alternatives for overcoming the constraints and implementing the requirements. Basic constraint in product development and testing is that product must be capable of satisfying customer needs. Needs are 'must' among requirements from customer's perspective. They may be part of processes for doing requirement analysis and selection of approach for designing on the basis of decision analysis and resolution, using some techniques such as cost-benefit analysis, etc. which suites best in the given situation. This must help organisation to achieve customer satisfaction through product development and delivery.

Meeting Customer Expectations Customer expectations may be categorised into two parts viz. expressed expectations and implied expectations. Expectations documented and given formally by the customer are termed as 'expressed requirements' while 'implied expectations' are those, which may not form a part of requirement specifications formally but something which is expected by customer by default. It is a responsibility of a developing organisation to convert, as many as possible, implied requirements into expressed requirements by asking queries or eliciting requirements. One must target for 100% conversion of implied requirements into expressed requirements, though difficult, as developer may refuse to accept the defect belonging to implied requirements simply because it is not a part of requirement statement and they may not be aware of such things.

Treating Every Customer with Integrity, Courtesy, and Respect Customer and the requirements assessed (both expressed as well as implied) are very important for a developing organisation as customer will be paying on the basis of value he finds in the product. Definition of the requirements is a first step to satisfy the customer through '**Conformance to Requirements**'. Requirements may be documented by anybody, generally development team, but customer is the owner of the requirements. Organisation shall believe and understand that the customer understands what is required by him. How to achieve these requirements is a responsibility of a developing organisation. He may be given suggestions, sharing some past experiences and knowledge gained in similar projects but manufacturer shall not define requirements or thrust or push the requirements to customer. It is quite often quoted that the customer does not know or understand his requirements. This opinion cannot be bought by anybody.

Customer telephone calls and mails must be answered with courtesy and in reasonable time. The information provided to the customer must be accurate and he/she must be able to depend on this information. The customer is not a hindrance to the project development but he is the purpose of the business and producer must understand this.

1.6.2 SUPPLIER'S VIEW OF QUALITY

Supplier is a development organisation in the context of software application development. Supplier has some expectations or needs, which must be satisfied by producing a product and selling it to customer. Supplier expectations may range from profitability, name in market, repeat orders, customer satisfaction, etc. These expectations may be fulfilled in the following ways,

Doing the Right Things Supplier is intended to do right things for the first time so that there is no waste, scrap, rework, etc. Wastes like rework, scrap, and repairs are produced by hidden factory for which there is no customer. Changes in requirements are considered as problems during product development, if supplier is expected to absorb the costs associated with it. Changes in requirement cause rework of design, development, testing etc. This adds to the cost of development but if the price remains same, it is a loss for manufacturer. It also adds to fatigue and frustration of the people involved in development as they find no value in reworking, sorting, scrapping, etc. Following right processes to get the product required by the customer and achieving customer satisfaction and profits as well as job satisfaction may be the expectations of a supplier. Suppliers may require clear and correct definition of deliverables, time schedules, attributes of product, etc.

Doing It the Right Way A producer may have his own methods, standards and processes to achieve the desired outputs. Sometimes, customer may impose the processes defined by him for building the product on the producer. Ability of development process to produce the product as required by the customer defines the capability of development process. These process definitions may be an outcome of quality standards or models or business models adopted by the supplier or customer. As organisation matures, the processes towards excellence—by refining and redefining processes and process capability—must improve continually continuously. As the processes reach optimisation, they must result into better quality products and more satisfaction for the customer and also fulfill vendor requirements.

Doing It Right the First Time Doing right things at the first time may avoid frustration, scrap, rework, etc. and improve profitability, reduce cost and improve customer satisfaction for the supplier. Doing right things at the first time improves performance, productivity and efficiency of a manufacturing process. This directly helps in improving profitability and gives advantage in a competitive market. Supplier would always like to follow the capable processes which can get the product right at the first attempt.

Doing It on Time All resources for developing a new product are scarce and time factor has a cost associated with it. The value of money changes as time changes. Thus, money received late is as good as less money received. If the customer is expected to pay on each milestone, then the producer has to deliver milestones on time to realise money on time. Delay in delivery represents a problem with processes, starting from getting requirements, estimation of efforts and schedule and goes upto delivery and deployment.

Difference between the two views discussed above (Customer's view vs Supplier's view) creates problem for manufacturer as well as customer when it comes to requirement gathering, acceptance testing, etc. It may result into mismatch of expectation and service level, which would be responsible for introducing defects in the product in terms of functionalities, features, cost, delivery schedule, etc. Sometimes these views are considered as two opposite sides of a coin or two poles of the Earth which can never match. The difference between two views is treated as a gap.

In many cases, the customer wants to dictate the terms as he is going to pay for the product and the processes used for making it, while the supplier wants customer to accept whatever is produced. Wherever the gap is less, the ability of a product to satisfy customer needs is considered better. At the same time, it helps a development organisation as well as a customer to get their parts of benefits from steady processes. On the contrary, larger gap causes more problems in development, distorted relations between two parties and may result into loss for both. Quality processes must reduce the gap between two views effectively. Effectiveness of a quality process may be defined as the ability of the processes and the product to satisfy both or all stakeholders in achieving their expectations.

Figure 1.1 explains a gap between actual product, requirements for the product and customer expectations from the product. It gives two types of gaps, viz. user's gap and producer's gap.

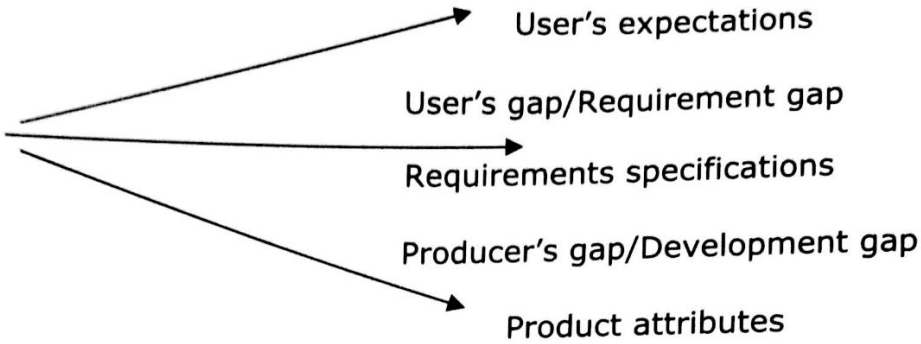

Fig. 1.1 User's gap and Producer's gap

1.6.3 USER'S GAP/REQUIREMENT GAP

User's gap is a gap between requirement specifications for the product and user expectations from it. This gap focuses on the difference in the final product attributes as defined by requirement statements with respect to the intents of the user. Developers must convert user needs into the product requirement specifications and create the product exactly as per these specifications. This may need understanding and interpretation of customer's business flow and requirements, and how the product is intended to be used by the customer to satisfy his business requirements.

Closing User's Gap User's gap represents the failure seen by the customer in terms of difference between user needs and product specifications. An organisation must apply some processes and methods so that user's gap can be closed effectively or reduced to as little as possible. Methods for closing these gaps may depend on organisation's way of thinking, resource availability, type of customer, customer's thought process, etc. Some of these methods are mentioned below.

Customer Survey Customer surveys are essential when an organisation is producing a product for a larger market where a mismatch between the product and expectation can be a major problem to producer. It may also be essential for projects undertaken for a single customer such as mission critical projects where failure of the project has substantial impact on the customer as well as producer. For a larger market, survey may be conducted by marketing function or business analyst to understand user requirements and collate them into specification documents for the product. Survey teams decide present and future requirements for the product and the features required by the potential customers.

For a single customer, a survey is conducted by business analyst and system analyst to analyse the intended use of the product under development and the possible environment of usage. Domain experts from the development side may visit the customer organisation to understand the specific requirements and workflow related to domain to incorporate it into the product to be developed.

Joint Application Development (JAD) Applications are developed jointly by customer and manufacturer where there is close co-ordination between two teams. In joint application development, users or customers may be overseeing the system development and closely monitor requirements specifications, architecture, designs, coding, testing and test results. The applications produced by this method may follow top-down approach where user interfaces and framework are developed first and approved by the users and then logic is build behind it. Some people may call joint application development as an agile methodology of development where developers collaborate with customer during development.

User Involvement in Application Development This approach works on the similar lines of joint application development (JAD). User may be involved in approving requirement specifications, design specification, application user interfaces, etc. He/she may have to answer the queries asked by developers and provide clarifications, if any. An organisation may develop a prototype, model, etc. to understand user requirements and get approval from the user team. If the organisation is producing products for a larger market, few representative users may be considered for collecting requirement specifications, test case designing and acceptance testing of the application under development.

1.6.4 PRODUCER'S GAP/DEVELOPMENT GAP

Producer's gap is a gap between product actually delivered and the requirement and design specifications developed for the product. The requirement specifications written by business analyst may not be understood in the same way by the development team. There are communication losses at each stage and business analyst and developers/testers may not be located next to each other to provide explanation for each and every requirement. More stages of communication may lead to more gaps and more distortion of requirements. The product so produced and requirement specifications used may differ significantly creating producer's gap.

Closing Producer's Gap Producer's gap represents a failure on part of development team to convert requirements into product. Producer's gap can be seen as the defects found in in-house testing. Producer's gap is due to process failure at producer's place and there must be process improvement plans to close this gap.

Process Definition Development and testing processes must be sufficiently mature to handle the transfer of information from one person or one stage to another during software development life cycle. There must be continuous 'Do' and 'Check' processes to build better products and get feedback about the process performance. Such product development has life cycle testing activities associated with development.

Work Product Review As the stages of software development life cycle progresses, one may have to keep a close watch on artifacts produced during each stage to find if any inconsistency has been introduced with respect to earlier phase. Generation of requirement traceability matrix is an important factor in this approach.

1.7 FINANCIAL ASPECT OF QUALITY

Earlier, people were of the opinion that more price of a product represents better quality as it involves more inspection, testing, sorting, etc. and ensures that only good parts are supplied to the customer. Sales price was defined as,

Sales price = Cost of manufacturing + Cost of Quality + Profit

If we consider the monopoly way of life, this approach may be considered good since the price is decided by the manufacturer depending upon three factors described above. Unfortunately, monopoly does not exist in real world. If any product enjoys higher profitability, more number of producers would enter into competition. Number of sellers may exceed number of buyers. When the products produced match in all aspects, the cost would decide the quantity of sale. The competitor who reduces the price, may get more volume of sale if all other things remain constant. Reducing the sales price reduces percentage profit. For maintaining profit, the producer may try to reduce cost of production without compromising on the quality aspect.

Thus, in a competitive environment, the equation changes to

Profit = Sales price – [Cost of manufacturing + Cost of Quality]

1.7.1 COST OF MANUFACTURING

Cost of manufacturing is a cost required for developing the right product by right method at the first time. The money involved in resources like material, people, licenses, etc. forms a cost of manufacturing. The cost of manufacturing remains constant over the time span for the given project and given technology and it has a direct relationship with the efforts. It can be reduced through improvements in technology and productivity but it may need longer time frame. The cost involved in requirement analysis, designing, development and coding are the costs associated with manufacturing.

1.7.2 COST OF QUALITY

Cost of quality represents the part of cost of production incurred in improving or maintaining quality of a product. Some people keep cost of manufacturing and cost of quality as separate while others may include them under cost of production. Cost of manufacturing may be supported by cost of quality and there exists an interrelationship between the two costs.

Cost of quality includes all the efforts and cost incurred in prevention of defects, appraisal of product to find whether it is suitable to customer or not and fixing of defects or failures at various levels as and when they are reported and conducting any retesting, regression testing, etc.

Cost of Prevention An organisation may have defined processes, guidelines, standards of development, testing, etc. It may define a program of imparting training to all people involved in development and testing. This may represent a cost of prevention. Creation and use of formats, templates, etc. acquiring various process models and standards, etc. also represent a cost of prevention. This is an investment by an organisation and it is supposed to yield returns. This is also termed as '**Green money**'. Generally, it is believed that 1 part of cost of prevention can reduce 10 parts of cost of appraisal and 100 parts of cost of failure.

Cost of Appraisal An organisation may perform various levels of reviews and testing to appraise the quality of the product and the process followed for developing the product. The cost incurred in first time reviews and testing is called as the cost of appraisal. There is no return on investment but this helps in identifying the process capabilities and process related problems, if any. This is termed as '**Blue money**' as it can be recovered from the customer. Generally, it is believed that 1 part of cost of appraisal can reduce 10 parts of cost of failure.

Cost of Failure Cost of failure starts when there is any defect or violation detected at any stage of development including post delivery efforts spent on defect fixing. Any extent of rework, retesting, sorting, scrapping, regression testing, late payments, sales under concession, etc. represents cost of failure. There may be some indirect costs such as loss of goodwill, not getting customer references, not getting repeat orders, and customer dissatisfaction associated with the failure to produce the right product. The cost incurred due to some kind of failure is represented as cost of failure. This is termed as '**Red money**'. This cost affects the profitability of the project/organisation badly.

On the basis of quality focus, organisations may be placed in 2 categories viz. organisations which are less quality conscious (termed as 'q') and organisations which are more quality conscious (termed as 'Q'). The distribution of cost of quality for these two types may be represented as below. Please refer Fi...

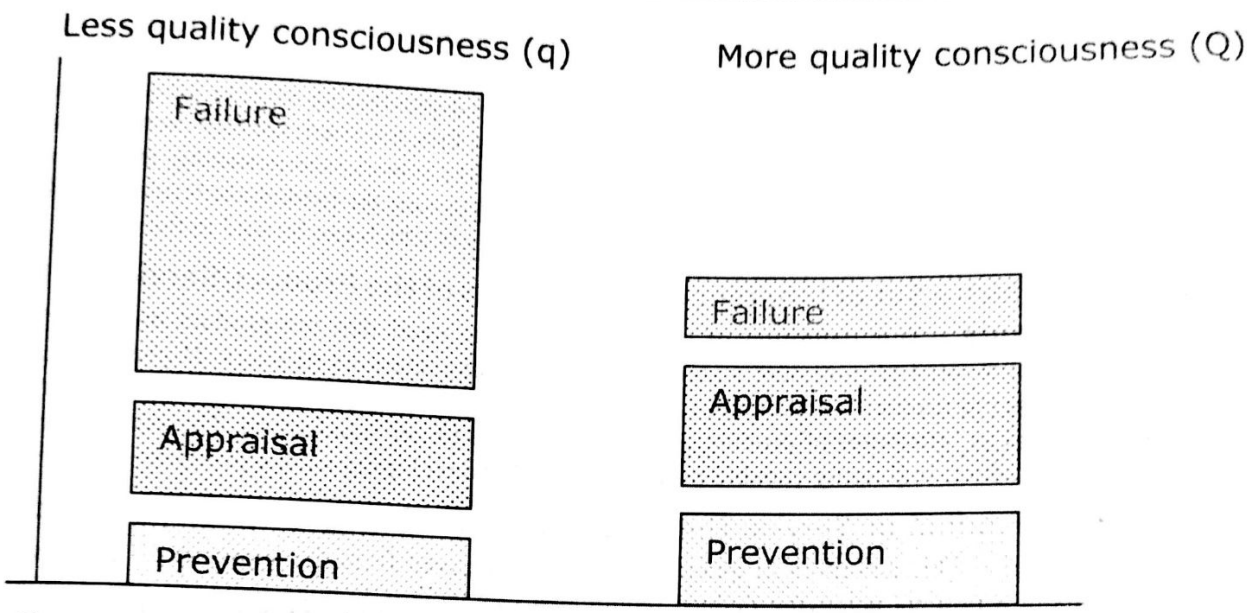

Fig. 1.2 Cost of Quality

The bottommost rectangle represents cost of prevention. As the organisation's quality consciousness increases, prevention cost increases substantially due to introduction of various process requirements. The organisation may have defined processes, methods, work instructions, standards, guidelines, templates, formats etc. Teams are supposed to use them while building a product or testing it, and conduct audits which need resources, time and money. Project teams may create project plan, test plan, assess the risks and issues faced during development, decide on the actions to reduce their probabilities/impacts, etc. More cost of prevention may be justifiable for a project/product only if it reduces the cost of failure.

Middle rectangle represents cost of appraisal. As quality consciousness increases, cost of appraisal also increases. An organisation may prepare various plans (such as quality plan and test plan) and then review these plans, write the test cases, define test data, execute test cases, analyse defects and initiate actions to prevent defect recurrence. There may be checklists, guidelines, etc. used for verification and validation activities. The cost incurred in appraisal must be justifiable and must reduce cost of failure.

The topmost rectangle represents cost of failure. Cost of failure includes the cost associated with any failure which may range from rework, retesting, etc. including customer dissatisfaction or loss of revenue, etc. As quality consciousness improves, failure cost must reduce representing better quality of products offered and higher customer satisfaction. Thus the overall cost of quality must reduce as quality consciousness of the organisation increases.

1.8 DEFINITION OF QUALITY

Let us try to redefine and understand the meaning of the term 'Quality' with a new perspective. Many definitions of the word 'Quality' are available and are used at different forums by different people. Most of these definitions show some aspect of quality. While no definition is completely wrong, no definition is completely right also. Few of the definitions prescribed for quality are,

- **Predictable Degree of Uniformity, Dependability at Low Cost and Suited to Market** This definition stresses on quality as an attribute of product which is predictable and uniform in

behavior throughout product usage and stresses on the ability of a product to give consistent results again and again. One must be able to predict the product behavior beforehand. It talks about reduction in variability of a product in terms of features, performance, etc. It also defines the dependability aspect of quality product. One must expect reliable results from quality products every time they are used. Quality product must be the cheapest one as it talks about reducing failure cost of quality like rework, scrap, sorting, etc. The most important thing about product quality is that it must help the product to suite the market needs or expectations. This necessitates that the manufacturer should produce those products which can be sold in the market.

- **Degree to Which a Set of Inherent Characteristics of the Product/Service Fulfills the Requirements** This definition of quality stresses the need that the product must conform to defined and documented requirement statement as well as expectations of users. Higher degree of fulfillments of these requirements makes a product better in terms of quality. There must be something in the product, defined as 'attributes of product', which ensures customer satisfaction. The attributes must be inborn in the product, indicating capable processes used for developing such a product.

- **Ability of a Product or Service That Bears Upon Its Ability to Satisfy Implied or Expressed Need** This definition of quality of product is derived from the approach of '**Fitness for use**'. It talks about a product achieving defined requirements as well as implied requirements, satisfying needs of customer for which it is being used. Better ability of a product to fulfill requirements makes a product better, from quality perspective.

1.9 CUSTOMERS, SUPPLIERS AND PROCESSES

For any organisation, there are some suppliers supplying the inputs required and some customers who will be buying the outputs produced. Suppliers and customers may be internal or external to the organisation. In the larger canvas, an entire organisation can be viewed as the component in a huge supply chain of the world where products are made by converting some inputs which may act as inputs to the next stage. External suppliers provide input to the organisation and external customers receive the output of the organisation. In turn, suppliers may be customers for some other organisations and customers may be acting as suppliers for somebody else down the line.

Internal Customer Internal customers are the functions and projects serviced and supported by some other functions/projects. System administration may have projects as their customer while purchasing may have system administration as their customer. During value chain, each function must understand its customers and suppliers. Each function must try to fulfill its customer requirements. This is one of the important considerations behind 'Total Quality Management' where each and every individual in supply chain must identify and support his customer. If internal customers are satisfied, this will automatically satisfy external customer as it sets the tone and perspective for everybody.

External Customer External customers are the external people to the organisation who will be paying for the services offered by the organisation. These are the people who will be actually buying products from the organisation. As the organisation concentrates on external customer for their satisfaction, it must improve quality of its output.

1.10 TOTAL QUALITY MANAGEMENT (TQM)

'**Total quality management**' principle intends to view internal and external customers as well as internal and external suppliers for each process, project and for entire organisation as a whole. The process and factions

of an organisation can be broken down into component elements, which act as suppliers/customers to each other during the workflow. Each supplier eventually also becomes a customer at some other moment and vice versa. If one can take care of his/her customer (whether internal or external) with an intention to satisfy him, it may result into customer satisfaction and continual improvement for the organisation.

Supply chain relationship may be defined graphically as, shown in Fig. 1.3.

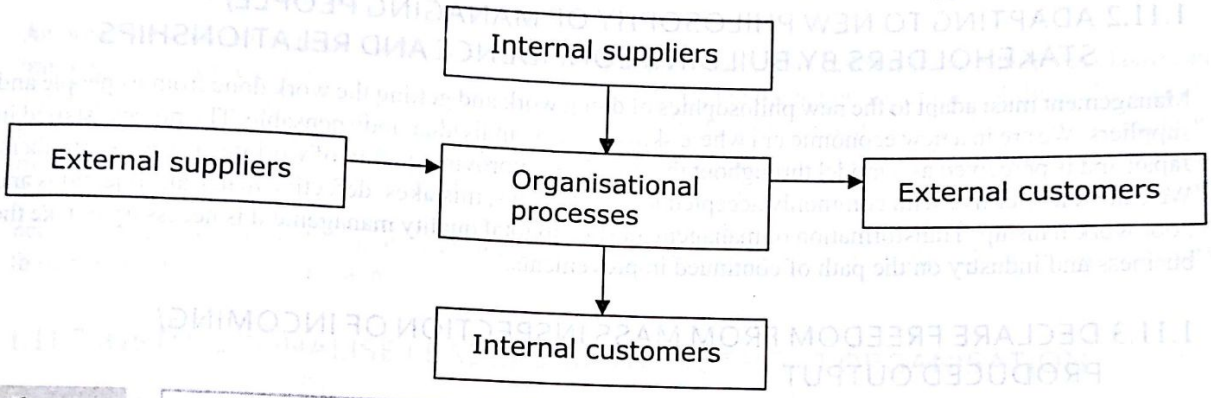

Fig. 1.3 Supply chain relationship between suppliers and customers

'Total quality management' (TQM) is the application of quality principles to all facets and business processes of an organisation. It talks about applying quality methods to the entire organisation whether a given function or part of the organisation faces external customer(s) or not. One clear definition of quality involves satisfying one's customer irrespective of whether he/she is outside or inside the organisation. This implementation of customer satisfaction has different meanings for different parts of an organisation.

Quality Management Approach Dr Edward Deming implemented quality management system driven by '**Total Quality Management**' and '**Continual Improvement**' in Japanese environment. It resulted into repetitive, cost effective processes with an intention to satisfy customer requirements and achieve customer satisfaction. Implementation was inclined toward assessment of quality management system which was adoptive to the utility of tools for understanding data produced by the process measurements. Dr Deming proposed principles for quality management that are widely used by the quality practitioners.

1.11 QUALITY PRINCIPLES OF 'TOTAL QUALITY MANAGEMENT'

'Total quality management' works on some basic principles of quality management definition and implementation. These have evolved over a span of experimentation and deployment of quality culture in organisations.

1.11.1 DEVELOP CONSTANCY OF PURPOSE OF DEFINITION AND DEPLOYMENT OF VARIOUS INITIATIVES

Management must create constancy of purpose for products and processes, allocating resources adequately to provide for long term as well as short term needs rather than concentrating on short term profitability: suppliers and organisation must have an intent to become competitive in the world, to stay in business and to provide jobs to people and welfare of the society. The processes followed during entire lifecycle of product

development from requirement capturing till final delivery must be consistent with each other and must be followed over a larger horizon. Decisions taken by management at different instances must be consistent and based upon same standards, rules and regulations. Different initiatives by management must have relationships with each other and must be able to satisfy the vision of the organisation.

1.11.2 ADAPTING TO NEW PHILOSOPHY OF MANAGING PEOPLE/ STAKEHOLDERS BY BUILDING CONFIDENCE AND RELATIONSHIPS

Management must adapt to the new philosophies of doing work and getting the work done from its people and suppliers. We are in a new economic era where skills make an individual indispensable. The process started in Japan and is perceived as a model throughout the world for improving quality of working as well as products. We can no longer live with commonly accepted levels of delays, mistakes, defective materials, rejections and poor workmanship. Transformation of management style to total quality management is necessary to take the business and industry on the path of continued improvements.

1.11.3 DECLARE FREEDOM FROM MASS INSPECTION OF INCOMING/ PRODUCED OUTPUT

It was a common belief earlier that for improving quality of a product, one needs to have rigorous inspection program followed by huge rework, sorting, scrapping, etc. It was believed that one must check everything to ensure that no defect goes to customer. But there is a need for change in thinking of management and people as mass inspection results into huge cost overrun and product produced is of inferior quality. There must be an approach for elimination of mass inspection followed by cost of failure as the way to achieve quality of products. Improving quality of products requires setting up the right processes of development and measurement of process capabilities and statistical evidence of built-in quality in all departments and functions.

1.11.4 STOP AWARDING OF LOWEST PRICE TAG CONTRACTS TO SUPPLIERS

Organisations must end the practice of comparing unit purchase price as a criterion of awarding contracts. Vendor selection must be done on the basis of total cost including price, rejections, etc. Organisations must perform measurements of quality of supply along with price and do the source selection on the basis of final cost paid by it in terms of procurement, rework, maintenance, operations, etc. It must reduce the number of suppliers by eliminating those suppliers that do not qualify with statistical evidences of quality of their supply. This will automatically reduce variation and improve consistency. Aim of vendor selection is to minimise total cost, not merely initial cost of purchasing, by minimising variations in the vendor supplied product. This may be achieved by moving towards lesser/single supplier, on a long term relationship of loyalty and trust. Organisations must install statistical process control in development even at vendor site and reduce the variation in place of inspecting each and every piece.

1.11.5 IMPROVE EVERY PROCESS USED FOR DEVELOPMENT AND TESTING OF PRODUCT

Improve every process of planning, production and service to the customer and other support processes constantly. Processes have interrelationships with each other and one process improvement affects other processes positively. Search continually for problems in these processes in terms of variations in order to

improve every activity in the organisation—to improve quality and productivity and decrease the cost of production as well as cost of quality continuously. Institutionalise innovations and improvements of products and processes used to make them. It is managements job to work continually for optimising processes.

1.11.6 INSTITUTIONALISE TRAINING ACROSS THE ORGANISATION FOR ALL PEOPLE

An organisation must institute modern methods of training which may include on-the-job training, classroom training, self study, etc. for all people, including management, to make better use of their abilities. New skills are required to keep up with the changes in materials, methods, product and service design, infrastructure, techniques and service. Skill levels of people can be enhanced to make them suitable for better performance by planning different training programs. Training—technical as well as nontechnical—should focus on improving present skills and acquiring new skills. Customer requirements may also change and people may need training to understand changes in customer expectations. Suppliers may be included in training programs to derive better output from them.

1.11.7 INSTITUTIONALISE LEADERSHIP THROUGHOUT ORGANISATION AT EACH LEVEL

An organisation must adopt and institute leadership at all levels' with the aim of helping people to do their jobs in a better way. Responsibility of managers and supervisors must be changed from controlling people to mentoring, guiding, and supporting them. Also, their focus should shift from number of work items to be produced to quality of output. Improvement in quality will automatically improve productivity by reducing scrap, inspection, rework, etc. Management must ensure that immediate actions are taken on reports of inherited defects, maintenance requirements, poor tools, fuzzy operational definitions and all conditions detrimental to quality of products and services offered to final users.

1.11.8 DRIVE OUT FEAR OF FAILURE FROM EMPLOYEES

An organisation must encourage effective two-way communication and other means to drive out fear of failure from the minds of all employees. Employees can work effectively and more productively to achieve better quality output when there is no fear of failure. People may not try new things if they are punished for failure. Management should not stop or discount feedback coming to them even if it is negative. Giving positive as well as negative feedback should be encouraged, and it must be used to perform SWOT analysis followed by actions. Fear is something which creates stress in minds of people, prohibiting them from working on new ways of doing things. Fear can cause disruption in decision-taking process which may result into excessive defense and also, major defects in the product. People may not be able to perform under stress.

1.11.9 BREAK DOWN BARRIERS BETWEEN FUNCTIONS/DEPARTMENTS

Physical as well as psychological breaking down of barriers between departments and staff areas may create a force of cohesion. People start performing as a team and there is synergy of group activities. People in different areas must work as a team to tackle problems that may be encountered with products and customer satisfaction. Ultimate aim of the organisation must be to satisfy customers. All people working together and helping each other to solve the problems faced by customers can help in achieving this ultimate aim.

1.11.10 ELIMINATE EXHORTATIONS BY NUMBERS, GOALS, TARGETS

Eliminate use of slogans, posters and exhortations of the work force, demanding 'Zero Defects' and new levels of productivity, without providing methods and guidance about how to achieve it. Such exhortations create adverse relationships between supervisors and workers. Main cause of low quality and productivity is in the processes used for production as the numbers to be achieved may be beyond the capability of the processes followed by the workers. This may induce undue frustration and stress and may lead to failures. The Organisation shall have methods to demonstrate that the targets can be achieved with smart work.

1.11.11 ELIMINATE ARBITRARY NUMERICAL TARGETS WHICH ARE NOT SUPPORTED BY PROCESSES

Eliminate work standards that prescribe quotas for the work force and numerical goals for managers to be achieved. Substitute the quotas with mentoring and support to people, and helpful leadership in order to achieve continual improvement in quality and productivity of the processes. Numerical goals should not become the definition of achievement/targets. There must be a methodology to define what achievement is and what must be considered as a stretched target.

1.11.12 PERMIT PRIDE OF WORKMANSHIP FOR EMPLOYEES

Remove the barriers that take away the pride of workmanship for workers and management. People must feel proud of the work they are doing, and know how they are contributing to organisational vision. This implies complete abolition of the annual appraisal of performance and of 'management by objective'. Responsibility of managers and supervisors must be changed from sheer numbers to quality of output. Management must understand 'managing by facts'.

1.11.13 ENCOURAGE EDUCATION OF NEW SKILLS AND TECHNIQUES

Institute a rigorous program of education and training for people working in different areas and encourage self-improvement programs for everyone. What an organisation needs is not just good people; it needs people who can improve themselves with education to accept new challenges. Advances in competitive position will have their roots in knowledge gained by people during such trainings.

1.11.14 TOP MANAGEMENT COMMITMENT AND ACTION TO IMPROVE CONTINUALLY

Clearly define top management's commitment to ever-improving quality and productivity and their obligation to implement quality principles throughout the organisation. It is not sufficient that the top management commits for quality and productivity but employees must also see and perceive their commitment. They must know what it is that they are committed to—i.e., what they must do in order to show their commitment.

1.12 QUALITY MANAGEMENT THROUGH STATISTICAL PROCESS CONTROL

Dr Joseph Juran is a pioneer of statistical quality control with a definition of improvement cycle through Define, Measure, Monitor, Control and Improve (DMMCI). One must understand the interrelationships among

customers, suppliers and processes used in development, testing, etc. and establish quality management based on metrics program. There are three parts of the approach, namely,

1.12.1 QUALITY PLANNING AT ALL LEVELS

Quality is not an accident but is a result of deliberate effort to achieve something which is defined in advance. An organisation must have a definition of what they wish to achieve. Quality improvement must be planned at all levels of organisation and then only it can be achieved. Quality planning happens at two levels viz. organisation level and individual department function project level.

- *Quality Planning at Organisation Level* Quality must be planned at organisation level first. It must be in the form of policy definition and strategic quality plans on the basis of vision, mission(s) and policies set by senior management. Planning process must attempt to discover who the customers are at present and who will be the customers in future, and what are and will be their needs and expectations from the organisation. The needs of the present or future customers must be expressed in numeric terms so that the actions can be planned and progress can be measured. The presence or absence of different attributes to the extent required shall define the quality level of the product or services.

- *Quality Planning at Unit Level* Quality planning at unit level must be done by the people responsible for managing the unit. Operational quality plans must be in sink with organisational policies and strategies. Project plan and quality plan at unit level must be consistent with the strategic quality plans at organisation level and must be derived from the organisation's vision and mission(s).

1.12.2 QUALITY CONTROL

Quality control process attempts to examine the present product at various levels with the defined standards so that an organisation may appraise the outcome of the processes. Removing defects in the processes to improve their capability can help to reach new levels of improved quality. (e.g., process improvement must be targeted towards lowering defects, reducing cost, and improving customer satisfaction.) It must measure the deviations with respect to the number of achievements planned in quality planning so that the organisation can initiate the actions to reduce the deviations to minimum level.

1.12.3 QUALITY IMPROVEMENT

Improvement process attempts to continuously improve the quality of the process used for producing products. Quality of the process is measured on the basis of the attributes of the products produced. There is no end to quality improvements and it needs to take newer challenges again and again. Finding deviations in the attributes of products and processes with respect to planned levels and permissible tolerances available shall guide the organisation to find the weak areas where actions may be prioritised.

1.13 QUALITY MANAGEMENT THROUGH CULTURAL CHANGES

Philip Crosby's approach to quality improvement is based on cultural change in an organisation towards total quality management. Quality management through cultural change defines quality improvements as a cultural change driven by management. It involves,

- Identifying areas in which quality can be improved depending upon process capability measurements and organisational priorities. An organisation must setup crossfunctional working groups (quality circles or

quality improvement teams) and try to improve awareness about the customer needs, quality and process measurements. The organisation may not be able to improve in all areas at a time and prioritisation may be essential depending upon some techniques such as Pareto analysis, Cost benefit analysis, etc. It must prioritise the improvements depending upon the resources available and efforts/investments required and the benefits derived from such improvements.

- Instituting teams representing different functions and areas for quality improvement can help in setting the change of culture. Improving quality of the processes of development, testing, managing, etc. is a team work led by management directives. A single person may not be able to institutionalise improvements across the organisation. Improvements in processes automatically improve the product and customer satisfaction.
- Setting measurable goals in each area of an organisation can help in improving processes at all levels. Goals may act as stretched targets with respect to what is currently achieved by the organisation. Goals may be set with reference to customer expectations or something which may give competitive advantage to the organisation in the market.
- Giving recognition to achievers of quality goals will boost their morale and set a positive competition among the teams leading to organisational improvements. This can lead to dramatic improvements in all areas. Management must demonstrate commitment to quality improvement, and recognition of achievements is a step in this direction.
- Repeating quality improvement cycle continuously by stretching goals further for next phase of improvements is required to maintain and improve the status further. The organisation must evaluate the goals to be achieved in short term, long term, and the combination of both to realise organisational vision.

1.14 CONTINUAL (CONTINUOUS) IMPROVEMENT CYCLE

Plan, Do, Check, and Act (PDCA) Cycle Continual (Continuous) improvement cycle is based on systematic sequence of Plan–Do–Check–Act activities representing a never ending cycle of improvements. PDCA cycle was initially implemented in agriculture. It was implemented later in the electronic industry. TQM has made the PDCA cycle famous in all industries. PDCA improvement cycle can be thought of as a wheel of improvement continually (continuously) rolling up the problem-solving hill and achieving better and better results for the organisation in each iteration. Stages of continual (Continuous) improvement through PDCA cycle are,

Plan An organisation must plan for improvements on the basis of its vision and mission definition. Planning includes answering all questions like who, when, where, why, what, how, etc. about various activities and setting expectations. Expected results must be defined in quantitative terms and actions must be planned to achieve answers to these questions. Quality planning at unit level must be in sync with quality planning at organisation level. Baseline studies are important for planning. Baseline studies define where one is standing and vision defines where one wishes to reach.

Do An organisation must work in the direction set by the plan devised in earlier phase for improvements. Plan is not everything but a roadmap. It sets the direction but execution is also important. Actual execution of a plan can determine whether the results as expected are achieved or not. Plan sets the tone while execution makes the plan work. 'Do' process need inputs like resources, hardware, software, training, etc. for execution of a plan.

Check An organisation must compare actual outcome of 'Do' stage with reference or expected results which are planned outcomes. It must be done periodically to assess whether the progress is in proper direction or not, and whether the plan is right or not. Expected and actual results must be in numerical terms, and compared at some periodicity as defined in the plan.

Act If any deviations (positive or negative) are observed in actual outcome with respect to planned results, the organisation may need to decide actions to correct the situation. The actions may include changing the plan, approach or expected outcome as the case may be. One may have to initiate corrective and/or preventive actions as per the outcome of 'Check'. When expected results and actuals match with given degree of variation, one may understand that the plan is going in the right direction. Running faster or slower than the plan will need action. Figure 1.4 shows diagramatically PDCA cycle of continual improvement.

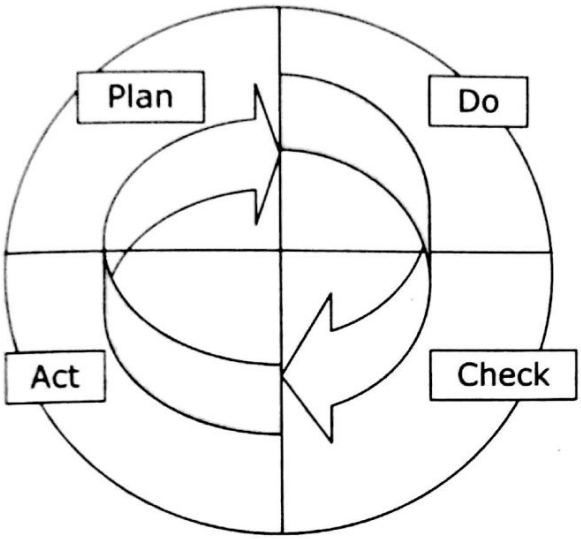

Fig. 1.4 Continual improvement cycle (PDCA cycle)

1.15 QUALITY IN DIFFERENT AREAS

Let us try to understand quality attributes of various products in different areas. Different domains need different quality factors. They may be derived from the customers/users of the domains. Here are few examples of some domains showing customer expectations in terms of quality for various products. These are generic expectations of customers in certain areas and may differ for some individual examples depending upon specific requirements. Definition of quality expectations will vary from instance to instance depending on the domain under consideration, type of product, type of customer, other competitive products, and their features. The table below lists different areas representing different domains and indicates some factors that might be considered related to quality in these areas. Table 1.2 shows some common expectations from customers.

Table 1.2 Products and expected attributes

Product/Service category	Expected attributes
Airlines Industry	On time arrival and departure, comfortable journey, low cost service, reliability and safety.
Health Care Industry	Correct diagnosis and treatment, minimum wait time, lower cost, safety and security.
Food Service Industry	Good product, good taste, fast delivery, good ambience, clean environment.
Consumer Products Industry	Properly made to suite individuals, defect free products, cost effective.
Military Services	Rapid deployment, decreased wages and cost, security.
Automotive Industry	Defect free product, less fuel consumption, more power, safe journey.
Communications Industry	Clear communications, faster access, cheaper service.

There are some common denominators in all these examples which may be considered as the quality factors. Although the terms used to explain each product in different domain areas vary to some extent, almost all areas can be explained in terms of few basic quality parameters. These are defined as the basic quality parameters by stalwarts of quality management.

- Cost of the product and value which the customer finds in it
- Service offered to the customers, in terms of support by the manufacturer
- Time required for the delivery of product
- Customer satisfaction derived from the attributes and functionalities of a product
- Number of defects in the product or frequency of failures faced by users

1.16 BENCHMARKING AND METRICS

Benchmarking is an important concept used in Quality Function Deployment (QFD). It is the concept of creating qualitative/quantitative metrics or measurable variables, which can be used to assess product quality on several scales against a benchmark. Typical variables of benchmarking may include price of a product paid by customer, time required to acquire it, customer satisfaction, defects or failures, attributes and features of product, etc. **Metrics** are defined for collecting information about the product capabilities, process variability and outcome of the process in terms of attributes of products. Metric is a relative measurement of some parameters of a product which are related to the product and processes used to make it. An organisation must develop consistent set of metrics derived from it's strategic business plan and performance of benchmark partner.

1.17 PROBLEM SOLVING TECHNIQUES

Improving quality of products and services offered to customers requires methods and techniques of solving problems associated with development and processes used during their lifecycle. An organisation must use metrics approach of process improvement because it needs to make quantitative measurements. These measurements can be used for problem solving using quantitative techniques.

Problem solving can be accomplished by both qualitative and quantitative methods but problem definition becomes easier when we put them against some measures or comparators.

- Qualitative problem solving refers to understanding a problem solution using only qualitative indexes such as high, medium, low, etc. depending on whether something is improving or deteriorating from the present status and so forth. This is a typical scenario for low maturity organisations where the problems are much broader and can be classified in different bands very easily. For initial stages of improvement, qualitative problem solving is sufficient to get faster results. It saves time in defining and measuring data accurately and basic maturity can be achieved.
- Quantitative problem solving requires specification of exact measures in numerical terms such as 'the cost has increased 32.5% during the last quarter' or 'the time required to produce one product unit is reduced by 32 minutes'. For highly matured organisations, quantitative analysis is required for further improvements as basic improvements are already done. It must follow the cycle of Define, Measure, Monitor, Control and Improve. Measurement of processes and products may need good measuring instruments with high level of accuracy and repeatability.

Given quantitative data, one can use statistical techniques to characterise a process. Quantitative methodologies make it possible to analyse and visualise what is actually happening in a process. Process variations can be understood in a better way and actions can be initiated to reduce the variability.

1.18 PROBLEM SOLVING SOFTWARE TOOLS

While buying software for data management and statistical analysis, many organisations find it to be a big investment in terms of money, resources, etc. One must answer the question 'Why should one use software tools to solve problems about quality?' There are some advantages and disadvantages associated with usage of such tools for problem solving.

Advantages of Using Software Tools for Analysis and Decision Making

- Accuracy and speed of the tools is much higher compared to performing all transactions and calculations manually. Calculations can form the basis for making decisions and hence should be as accurate as possible.
- Decision support offered by the tool is independent of personal skills and there is least variation from instance to instance. Tools can support in some fixed range depending upon its logic. Some tools can learn things and use them as required.
- Tools can implement theoretical means of assessing metrics about quality as defined by business law. There is no manual variation.
- Tools alleviate the hard work required to perform hand or calculator driven computations and give more accurate and faster results.
- Tools can be integrated with other systems to provide a systematic and highly integrated means of solving problems

Disadvantages of Using Computer Tools for Analysis and Decision Making

- These programs and tools need training before they can be used. Training incurs cost as well as time. Some tools need specific trainings to understand them and use them.
- All softwares/hardwares are prone for defects and these tools are not exceptions to it. There can be some mistakes while building/using them. Sometimes these mistakes can affect the decisions drastically.
- Decision has to be taken by human being and not by the tool. Tools may define some options which may be used as guide. Some tools can take decision in the limited range.
- Tools may mean more cost and time to learn and implement. Every tool has a learning curve.

1.18.1 TOOLS

Tools are an organisations analytical asset that assist in understanding a problem through data and try to indicate possible solutions. Quality tools are more specific tools which can be applied to solving problems faced by projects and functional teams while improving quality in organisations. Tools may be hardware/ software and physical/logical tools. We will learn more about quality tools in Chapter 16 on 'Qualitative and Quantitative Analysis'.

1.18.2 TECHNIQUES

Techniques indicate more about a process used in measurement, analysis and decision making during problem solving. Techniques are independent of tools but they drive tool usage. Techniques do not need tools for application while tools need techniques for their use. Same tool can be used for different purposes, if the techniques are differed. Table 1.3 gives a difference between tools & techniques.

Table 1.3 Difference between tools and techniques

Tools	Techniques
Usage of tool is guided by the technique. Tool is of no use unless technique (to use it) is available.	Technique is independent of any tool.
Different techniques may use the same tool to achieve different results.	Same technique may use different tools to achieve the same result.
Tool improvement needs technological change.	Technique change can be effected through procedural change.
Contribution of tools in improvement is limited.	Contribution of techniques in improvement is important.

Quality tips

- Try to define quality perspective for the organisation and set of products and projects executed by it.
- Define customer expectations rather than going for system requirement specifications.
- Assess the cost spent by an organisation under various heads of quality. Testers have to play a significant role in reducing cost of quality and improving profitability of the organisation.
- Understand and improve every aspect of an organisation through approaches, techniques and tools to enhance customer satisfaction and goodwill for the organisation. This can help the organisation to prosper in the long term.

Summary

In this chapter, we have seen various definitions of quality as understood by different people and different stakeholders. It also covered the definitions by quality stalwarts and different international standards. Then, we studied the basic components to produce quality, and the views of customers and producer on quality. As a tester, one must understand different gaps like user gap, and producer gap, and how to close them to achieve customer satisfaction.

We have described various cost components like manufacturing cost and cost of quality. Cost of quality concepts with its three components viz. preventive cost of quality, appraisal cost of quality and failure cost of quality are described along with the importance of cost of prevention, and how it affects cost of quality and improves profitability.

We have seen different approaches to continually (continuously) improving quality. We have covered 'TQM principles of quality management', and 'DMMCI principles of continual improvement' through quality planning, quality control and quality improvement. We have also discussed a theory of 'Cultural change principles of quality management'.

Finally we have introduced the concept of problem solving through usage of tools and techniques. We have also briefly elucidated the concept of benchmarking.

This chapter focusses on software quality parameters. It provides a basic understanding that quality expectation for different products is different and lays a foundation of quality management approach.

2.1 INTRODUCTION

In the previous chapter, we have discussed various definitions of quality and how they fit the perspective of different stakeholders. One of the definitions i.e., '**Conformance to explicitly stated and agreed functional and non-functional requirements**', may be termed as 'quality' for the software product offered to customers/final users from their perspective. Customers may or may not be the final user and sometimes, developers have to understand requirements of final users in addition to the customer. If the customer is a distributor or retailer or facilitator who is paying for the product directly, then the final user's requirements may be interpreted by the customer to make a right product.

It is not feasible to put all requirements in requirement specifications. A large number of requirements remain undefined or implied as they may be basic requirements for the domain and the customer perceives them as basic things one must know. It gives importance to the documented and approved software and system requirement specifications on which quality of final deliverables can be decided. It shifts the responsibility of making a software specification document and getting it approved from customer to producer of a product. There are many constraints for achieving quality for software application being developed using such software requirement specifications.

2.2 CONSTRAINTS OF SOFTWARE PRODUCT QUALITY ASSESSMENT

Generally, requirement specifications are made by business analysts and system analysts. Testers may or may not have direct access to the customer and may get information through requirement statements, queries answered, etc. either from the customer or business system/analyst, etc. There are few limitations of product quality assesment in this scenario.

- Software is virtual in nature. Software products cannot be seen, touched or heard. Our normal senses and measuring instruments are not capable of measuring quality of software, which is possible in most of the other engineering products.
- There is a huge communication gap between users of software and developers/testers of the product. Generally 5–8 agencies are involved between these two extreme ends. If an average communication loss of 10% at each stage is considered, then huge distortion is expected from user's perspective of requirements and actual product.
- Software is a product which is unique in nature. Similarities between any two products are superficial ones. The finer requirements, designs foundation, architecture, actual code, etc. may be completely different for different products. Way of software design, coding, and reusability may differ significantly from product to product though requirements may look similar at a global level.
- All aspects of software cannot be tested fully as number of permutations and combinations for testing all possibilities tend to infinity. There are numerous possibilities and all of them may not be tried in the entire life cycle of a software product in testing or even in usage. Exhaustive testing is neither feasible nor justifiable with respect to cost.
- A software program executes in the same way every time when it is executing some instruction. An application with a problematic code executes wrongly every time it is executed. It makes a very small defect turn into a major one as the probability of defect occurrence is 100% when that part is executed.

Business analysts and system analysts must consider the following while capturing the requirements of a customer

2.3 CUSTOMER IS A KING

The customer is the most important person in any process of developing a product and using it—software development is not an exception. All software life cycle processes such as development, testing, maintenance, etc. are governed by customer requirements, whether implied or expressed.

An organisation must be dedicated to exceed customer satisfaction with the latter's consent. Exceeding customer satisfaction must not be received with surprise by the customer. He must be informed about anything that has been provided extra and must be in a position to accept it or reject it. Any surprise may be considered as defect.

A satisfied customer is an important achievement for an organisation and is considered as an investment which may pay back in short as well as long term (in terms of references, goodwill, repeat order, etc.). Satisfied customers may give references to others and come back with repeat orders. Customer references are very important for developing new accounts. Organisations should try to implement some of the following measures to achieve customer satisfaction.

2.3.1 FACTORS DETERMINING SUCCESS

To be a successful organisation, one must consider the following factors, entities, and their interactions with each other.

- **Internal Customer and Internal Supplier** 'Internal customer satisfaction' philosophy is guided by the principles of 'Total Quality Management'. When an organisation is grouped into various functions/

departments, then one function/department acts as a supplier or a customer of another function/department. If every function/department identifies its customers and suppliers and tries to fulfill their requirements, in turn, external customer requirements get satisfied.

- **External Customer and External Supplier** External customers may be the final users, purchasers of software, etc. Software requirement specifications are prepared and documented by developing organisations to understand what customer/final user is looking for when he wishes to acquire a particular product. Customers are actually paying for getting their needs served and not for the implementation of the requirements as defined in the specifications document. External suppliers are the entities who are external to the organisation and who are supplying to the organisation.

2.4 QUALITY AND PRODUCTIVITY RELATIONSHIP

Many people feel that better quality of a product can be achieved by more inspection or testing, reworking, scrapping, sorting, etc. More inspection cycles mean finding more defects, fixing defects mean better quality (as it will expose maximum possible defects to be fixed by developers), and ultimately, the customer may get a better product. This directly means that there would be more rework/scrap and it will lead to more cost/price or less profit for such products as more effort and money is spend in improving quality. In such cases, time and effort required would be much higher.

In reality, quality improvement does not talk about product quality only but a process quality used for making such a product. If the processes of development and testing are good, a bad product will not be manufactured in the first place. It will reduce inspection, testing, rework, and cost/price. Thus quality must improve productivity by reducing wastage. It must target for doing 'first-time right.'

- **Improvement in Quality Directly Leads to Improved Productivity** Improved quality does not mean more inspection, testing, sorting and rejection but improving the processes related to product development. All products are the outcome of processes, and good processes must be capable of producing good product at the first instance. This approach reduces frustration, rejection, rework, inspection, and improves quality and customer satisfaction. As this hidden factory producing scrap and wastage stops working, productivity and efficiency improves. Thus, quality products must give more profitability to the supplier and is a cheaper option for the customer.

- **The Hidden Factory Producing Scrap, Rework, Sorting, Repair, and Customer Complaint is Closed** Customer does not intend to pay for scrap, rework, sorting, etc. to get a good product. Engineering industry has faced this problem of unwillingness of customer to pay even for first-time inspection/testing as they represent deficiencies in manufacturing processes. Customer complaints are mostly due to the problems associated with products and aligned services. Problems in products can be linked to faulty development processes. Either the defects are not found in the product during process of development or testing, or fixing of the defects found in these processes introduces some more defects in the product offered.

- **Effective Way to Improve Productivity is to Reduce Scrap and Rework by Improving Processes** Productivity improvement means improving number of good parts produced per unit time and not the parts produced per unit time. It is not hard work but smart and intelligent work which can

help an organisation in improving product quality, productivity and customer satisfaction by reducing rework, scrap, etc. It necessitates that an organisation must incorporate and improve quality in the processes which lead to better product quality. As wastage of resources reduce with improvements in quality, productivity and efficiency improves as a direct result by improvements in processes.

- **Quality Improvements Lead to Cost Reduction** Quality improves productivity, efficiency and reduces scrap, rework, etc. Improvement in quality increases profit margins for producer by reducing cost of development, cost of quality and reduces sales price for customer. Thus quality implementation must reduce the cost and price without sacrificing profitability.

Employee Involvement in Quality Improvement Employee is the most important part of quality improvement program and crucial element for organisational performance improvement. Management leadership and employee contribution can make an organisation quality conscious while lack of either of the two can create major problems in terms of customer dissatisfaction, loss of profitability, loss of goodwill, etc. Employees are much nearer to problematic processes and know what is going wrong and how it can be improved. Employee involvement in quality implementation is essential as the employees facing problems in their work indicate the process problems. Management must include employees in quality improvement programs at all stages.

- **Proper Communication Between Management and Employee is Essential** Communication is a major problem in today's environment. One of the communication hurdles is that the chances of face-to-face communication are reducing due to technological improvements and distributed teams. Mostly, communication is by virtual appliances like emails, video conferencing, etc. where it is difficult to judge the person through body language. Either there is no communication or there is excessive communication leading to a problematic situation. There are huge losses in communication and distortions leading to miscommunication and wrong interpretation. The reasons for 'Producer's gap' and 'User's gap' are mainly attributed to communication problems. Different words and terms used during the message, tone, type of message, speaking skills, listening skills, etc. contribute to quality of communication.

- **Employees Participate and Contribute in Improvement Process** In quality improvement program design and implementation, employees perform an important part of identification of problems related to processes and giving suggestions for eliminating them. They are the people doing actual work and know what is wrong and what can be improved in those processes to eliminate problems. They must be closely associated with the organisation's goal of achieving customer satisfaction, profitability, name and fame in market, etc. Every employee needs to play a part in implementation of 'Total Quality Management' in respective areas of working. This can improve the processes by reducing any waste.

- **Employee Shares Responsibility for Innovation and Quality Improvement** Management provides support, guidance, leadership, etc. and employees contribute their part to convert organisations into performing teams. Everyday work can be improved significantly by establishing small teams for improvement which contribute to innovations. An organisation must not wait for inventions to happen for getting better products but perform small improvements in the processes everyday to achieve big targets in the long range. The theory of smart work in place of hard work helps employees to identify any type of waste in the process of development and eliminate it.

Table 2.1 Difference between inventions and innovation

Invention	Innovation
Inventions may be accidental in nature. They are generally unplanned.	Innovation is a planned activity leading to changes.
Invention may or may not be acceptable to people doing the work immediately. Inventions are done by scientist and implementation and acceptance by people can be cumbersome as general level of acceptance is very low.	Innovation is done by people in a team, possibly cross-functional teams, involved in doing a work. There is higher acceptability by people as they are involved in it.
Inventions may not be directly applied to everyday work. It may need heavy customisation to make it suitable for normal usage.	Innovations can be applied to every day work easily. The existing methods and processes are improved to eliminate waste.
Breakthrough changes are possible due to inventions.	Changes in small steps are possible by innovation.
Invention may lead to major changes in technology, way of doing work, etc.	Innovation generally leads to administrative improvements, whereas technological or breakthrough improvements are not possible.
Invention may lead to scraping of old technologies, old skills and sometimes, it meets with heavy resistance.	Innovation may lead to rearrangement of things but there may not be a fundamental change. It generally works on elimination of waste.

Table 2.1 gives a difference between invention & innovation.

Many organisations have a separate 'Research and Development' function responsible for doing inventions. These functions are dedicated to make breakthrough changes by developing new technologies and techniques for accomplishing work. They are supposed to derive major changes in the approaches of development, implementation, testing, etc. 'Six sigma' improvements also talk about breakthrough improvements where processes may be redesigned/redefined. It may add new processes and eliminate old processes, if they are found to be problematic. But, an organisation should also create an environment which helps in innovation or rearrangement of the tasks to make small improvements everyday. Continuously identifying any type of waste in day-to-day activities and removing all nonessential things can refine the processes.

2.5 REQUIREMENTS OF A PRODUCT

Everything done in software development, testing and maintenance is driven by the requirements of a product to satisfy customer needs. There are basic requirements for building software, which will help customers conduct their businesses in a better way. Every product offered to a customer is intended to satisfy some requirements or needs of the customer. Requirements may be put in different categories. Some of these are,

- **Stated/Implied Requirements** Some requirements are specifically documented in software requirement specifications while few others are implied ones. When we build software, there are functional and non-functional requirements specified by a customer and/or business/system analyst. It is also intended not to violate some generally accepted requirements such as 'No spelling mistakes in user interfaces', 'Captions on the control must be readable', etc. These type of requirements may not be documented as

a part of requirement statement formally but generally considered as requirement, and are expected in a product. As a part of development team/test team, one must understand stated as well as intended or implied requirements from the users. It may be a responsibility of a developing organisation to convert as many implied requirements as possible into stated requirements. Though impossible, the target may be 100%.

- **General/Specific Requirements** Some requirements are generic in nature, which are generally accepted for a type of product and for a group of users while some others are very specific for the product under development. Addition of two numbers should be correct is a generic requirement while the accuracy of 8 digits after decimal and rounding may be a very specific requirement for the application under development. Usability is a generic requirement in software for the intended user while authentication and messaging to users may be driven by specific requirements of an application. Many times, the generic requirements are considered as implied ones and those may not be mentioned in requirement specifications while specific requirements are stated ones as those are present in requirement specifications.

- **Present/Future Requirements** Present requirements are essential when an application is used in present circumstances while future requirements are for future needs which may be required after some time span. For projects, present as well as future requirements may be specifically defined by a customer or business/system analyst. A product development organisation may have to do further research to identify or extrapolate future needs of users. For banking software, today's requirements may be 5000 saving accounts, and the application may be running in client-server environment. But a future need may be 50 lakh saving accounts and the application may be running as a web application. 'How much future?' must be guided by the customer's vision as this may influence product cost. Definition of future has direct relationship with usable life of an application. Some people may use a software for 3 years while some other may be planning to use it for 30 years. The future requirements may change as per the expected life span of the software.

On the basis of priority of implementation from user's perspective, requirements may be categorised in different ways as follows:

- **'Must' and 'Must not' Requirements or Primary Requirements** 'Must' requirements are primary requirements for which the customer is going to pay for while acquiring a product. These are essential requirements and the value of the product is defined on the basis of the accomplishment of 'must' requirements. Generally these requirements have the highest priority in implementation. Not meeting these requirements can cause customer dissatisfaction and rejection of a product. These requirements may be denoted by priority 'P1' indicating the highest priority. It also covers 'Must not' requirements which must be absent in the product.

- **'Should be' and 'Should not be' Requirements or Secondary Requirements** 'Should be' requirements are the requirements which may be appreciated by the customer if they are present/absent and may add some value to the product. Customer may pay little bit extra for the satisfaction of these requirements but price of the product is not governed by them. Generally, these requirements are lower priority requirements than 'Must' requirements. These requirements may give the customer delight, if present and little disappointment, if absent. These requirements may be denoted by priority 'P2'. It also covers 'Should not' requirements.

- **'Could be' and 'Could not be' Requirements or Tertiary Requirements** 'Could be' requirements are requirements which may add a competitive advantage to the product but may not add much value in terms of price paid by a customer. If two products have everything same, then 'could be' requirements may help in better appreciation of a product by the users. These are the lowest priority requirements. It also covers 'Could not' requirements. These requirements give a product identity in the market. These requirements may be denoted by priority 'P3'.

While talking about a product in view of a bigger market with large number of generic users, it may be very difficult to categorise the requirements as mentioned above. This is because 'must' requirement for one customer may be 'could be' requirement for somebody else. In such cases, an organisation may have to target the customer segment and define the priorities of the requirements accordingly. Customer must have the final authority to define the category of requirement.

2.6 ORGANISATION CULTURE

An organisation has a culture based on its philosophy for existence, management perception and employee involvement in defining future. Quality improvement programs are based on the ability of the organisation to bring about a change in culture. Philip Crosby has prescribed quality improvement for cultural change. Quality culture of an organisation is an understanding of the organisations virtue about its people, customer, suppliers and all stakeholders. 'Q' organisations are more quality conscious organisations, while 'q' organisations are less quality conscious organisations. The difference between 'Q' organisations and 'q' organisation is enumerated as follows. Table 2.2 shows a difference between quality culture of 'Q' & 'q' organisations.

Table 2.2 Difference between 'Q' organisation and 'q' organisation

Quality culture is 'Q'	Quality culture is not 'q'
These organisations believe in listening to customers and determining their requirements.	These organisations assume that they know customer requirements.
These organisations concentrate on identifying cost of quality and focusing on it to reduce cost of failure which would reduce overall cost and price.	These organisations overlook cost of poor quality and hidden factory effect. They believe in more testing to improve product quality.
Doing things right for the first time and every time is the motto of success.	Doing things again and again to make them right is their way of working. Inspection, rework, scrap, etc. are essential.
They concentrate on continuous/continual process improvement to eliminate waste and get better output.	They work on the basis of finding and fixing the problem as and when it is found. Onetime fix for each problem after it occurs.
These organisations believe in taking ownership of processes and defects at all levels.	These organisations try to assign responsibility of defects to someone else.
They demonstrate leadership and commitment to quality and customer satisfaction.	They believe in assigning responsibility for quality to others.

2.6.1 SHIFT IN FOCUS FROM 'q' TO 'Q'

As the organisation grows from 'q' to 'Q', there is a cultural change in attitude of the management and employees towards quality and customer. In initial stages, at the level of higher side of 'q', a product is subjected to heavy inspection, rework, sorting, scrapping, etc. to ensure that no defects are present in final deliverable to the customer while the final stages of 'Q' organisation concentrate on defect prevention through

process improvements. It targets for first-time right. Figure 2.1 shows an improvement process where focus of quality changes gradually.

Fig. 2.1 Shift in focus from Quality control to Quality management

- **Quality Control Approach (Finding and Fixing Defects)** Quality control approach is the oldest approach in engineering when a product was subjected to rigorous inspection for finding and fixing defects to improve it. Organisations at the higher end of 'q' believe in finding and fixing defects to the extent possible as the way to improve quality of product before delivering it to customer and achieving customer satisfaction. It basically works on correction attitude involving defect fixing, scrap, rework, segregation, etc. It works on the philosophy that a product is good unless a defect is found in it. There are huge testing teams, large investment in appraisal cost and defect fixing costs followed by retesting and regression testing.

- **Quality Assurance Approach (Creation of Process Framework)** Quality assurance is the next stage of improvement from quality control where the focus shifts from testing and fixing the defects to first-time right. An Organisation does investment in defining processes, policies, methods for handling various functions so that it can incorporate a process approach for doing various things. It becomes a learning organisation as it shifts its approach from 'quality control' to 'quality assurance'. The management approach shifts from corrections to corrective actions through root cause analysis. There are some actions on the basis of metrics program instituted by the organisation. It also starts working on preventive actions to some extent to avoid potential defects. Defects are considered as failures of processes and not of people, and actions are initiated to optimise the processes.

- **Quality Management Approach** There are three kinds of system in the universe, viz. completely closed systems, completely open systems and systems with semipermeable boundaries. Completely closed systems represent that nothing can enter inside the system and nothing can go out of the system. On the other hand, open system represents a direct influence of universe on system and vice-a-versa. Completely closed systems or completely open systems do not exist in reality. Systems with semipermeable boundaries are the realities, which allow the system to get impacted from external changes and also have some effect on external environment. Anything coming from outside may have an impact on the organisation but the level of impact may be controlled by taking some actions. Similarly anything going out can also affect the universe but impact is controlled.

Organisations try to assess the impact of the changes on the system and try to adapt to the changes in the environment to get the benefits. They are highly matured when they implement Quality Management as a management approach. There are virtually no defects in processes as they are optimised continuously, and products are delivered to customers without any deficiency. The organisation starts working on defect prevention mechanism and continuous improvement plans. The organisation defines methods, processes and techniques for future technologies and training programs for process improvements.

Management includes planning, organising, staffing, directing, coordinating, and controlling to get the desired output. It also involves mentoring, coaching, and guiding people to do better work to achieve organisational objectives.

2.7 CHARACTERISTICS OF SOFTWARE

There are many products available in the market which are intended to satisfy same or similar demands. There is a vast difference between software products and other products due to their nature.

- Software cannot be sensed by common methods of inspection or testing, as it is virtual in nature. The product is in the form of executable which cannot be checked by any natural method available to mankind like touch, smell, hearing, taste, etc. It cannot be measured by some measuring instruments commonly available like weighing balance, scales, etc. It needs testing in real environment but nobody can do exhaustive testing by trying all permutations and combinations.
- There are different kinds of software products and their performance, capabilities, etc. vary considerably from each other. There are no same products though there may be several similar ones or satisfying similar needs. Every product is different in characteristics, performance, etc. Software is always unique in nature.
- Every condition defined by the software program gets executed in the same way every time when it gets executed. But the number of conditions, and algorithm combinations may be very large tending to infinity and testing of all permutations/combinations is practically impossible.

2.8 SOFTWARE DEVELOPMENT PROCESS

Software development process defines how the software is being built. Some people also refer to SDLC as system development life cycle with a view that system is made of several components and software is one of these components. There are various approaches to build software. Every approach has some positive and some negative points. Let us talk about few basic approaches of developing software from requirements. It is also possible that different people may call the same or similar approach by different names.

- Waterfall development approach/model
- Iterative development approach/model
- Incremental development approach/model
- Spiral development approach/model
- Prototyping development approach/model
- Rapid application development approach/model
- Agile development approach/model

2.8.1 WATERFALL DEVELOPMENT APPROACH/MODEL

Waterfall model is the simplest software development model and is used extensively in development process study. There are many offshoots of waterfall model such as modified waterfall model, iterative waterfall model, etc. Though it is highly desirable to use waterfall model, it may not be always feasible to work with it. Still, waterfall model remains as a primary focus for study purpose. It is also termed as classical view of software development as it remains the basis or foundation of any development activity. Most of the other models of development are based upon the basic waterfall model as it represents a logical way of doing things.

Typical waterfall model is shown in Fig. 2.2.

Arrows in the waterfall model are unidirectional. It assumes that the developers shall get all requirements from a customer in a single go. The requirements are converted into high level as well as low level designs. Designs are implemented through coding. Code is integrated and executables are created. Executables are

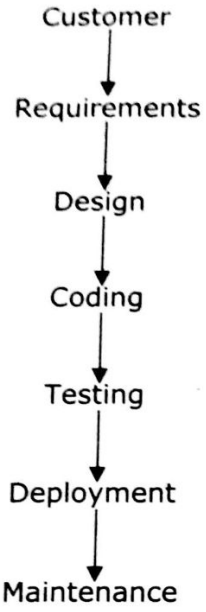

Fig. 2.2 Waterfall model

tested as per test plan. The final output in the form of an executable is deployed at customer premises. Future activities are handled through maintenance. If followed in reality, waterfall model is the shortest route model which can give highest efficiency, and productivity.

Waterfall models are used extensively in fixed price/fixed schedule projects where estimation is based on initial requirements. As the requirement changes, estimation is also revised.

Limitations of Waterfall Cycle There is no feedback loop available in waterfall model. It is assumed that requirements are stable and no problem is encountered during entire development life cycle. Also, no rework is involved in waterfall model.

2.8.2 ITERATIVE DEVELOPMENT APPROACH/MODEL

Iterative development process is more practical than the waterfall model. It does not assume that the customer gives all requirements in one go and there is complete stability of requirements. It assumes that changes may come from any phase of development to any previous phase and there are multiple permutations and combinations of changes. Changes may have a cascading effect where one change may initiate a chain reaction of changes. Figure 2.3 shows a feedback loop which is the fundamental difference between waterfall model and iterative development model.

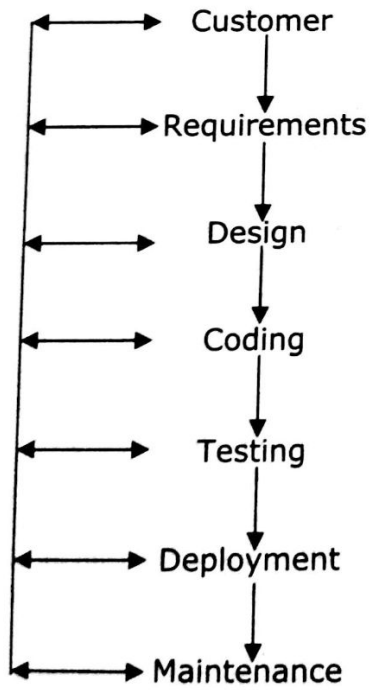

Fig. 2.3 Iterative development model

Limitations of Iterative Development Iterative development consists of many cycles of waterfall model. It gives problems in fixed price projects for estimation. Another problem faced by iterative development is that the product architecture and design becomes fragile due to many iterative changes.

2.8.3 INCREMENTAL DEVELOPMENT APPROACH/MODEL

Incremental development models are used in developing huge systems. These systems are made of several subsystems which in themselves are individual systems. Thus, incremental systems may be considered as a collection of several subsystems.

An individual subsystem may be developed by following waterfall methodology and iterative development. These subsystems may be connected to each other externally, either directly or indirectly. A directly interconnected system allows the subsystems

to talk with each other while indirectly interconnected system has some interconnecting application between two subsystems. Direct connectivity makes a system more robust but flexibility can be a major issue.

The incremental model gives flexibility to a customer. One system may be created and the customer may start using it. The customer can learn the lessons and use them while second part of the system is developed. Once those are integrated, third part may be developed and so on. The customer does not have to give all requirements at the start of development phase.

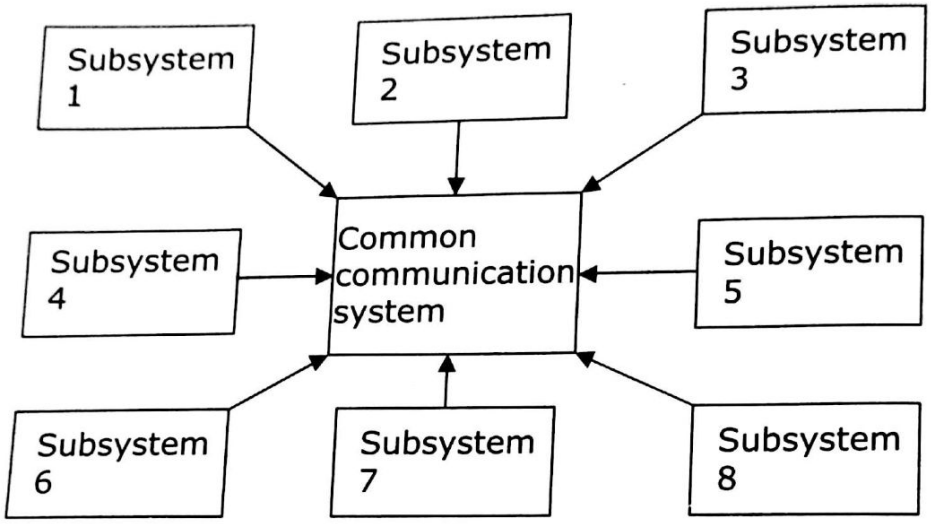

Fig. 2.4 Incremental model

The model in Fig. 2.4 shows a common communication system and incremental subsystems where different subsystems communicate through a common communication system.

Limitations of Incremental Development Incremental models with multivendor product integration are a major challenge as parameter passing between different systems may be difficult. Incremental models help in integration of big systems at the cost of loss of flexibility. When a system is incremented with new subsystems, it changes the architecture of that system. Increment in the system is followed by heavy regression testing to find that when multiple systems come together, can they work individually as well as collectively.

2.8.4 SPIRAL DEVELOPMENT APPROACH/MODEL

Spiral development process assumes that customer requirements are obtained in multiple iterations, and development also works in iterations. Many big software systems are built by spiral models of ever-increasing size. First some functionalities are added, then product is created and released to customer. After getting the benefits of first iteration of implementation, the customer may add another chunk of requirements to the existing one. Further addition of requirements increase the size of the software spirally. Sometimes, an individual part developed in stages represents a complete system, and it may communicate with the next developed system through some interfaces.

In many ERPs, initial development concentrated around material management part which later increased spirally to other parts such as purchasing, manufacturing, sales, warehousing, cash control, etc. Many banking softwares also followed a similar route. Figure 2.5 shows a spiral development model.

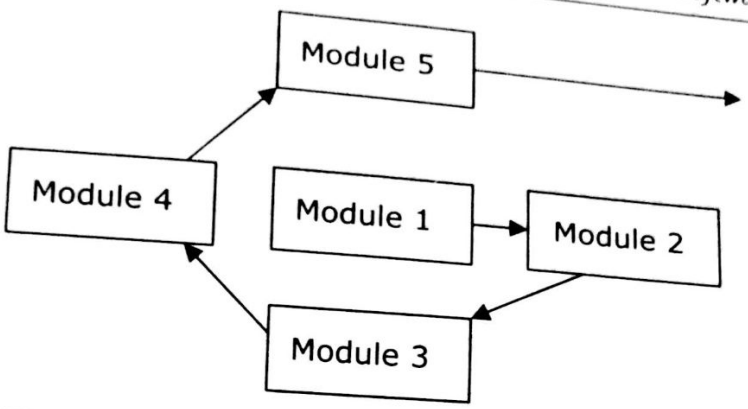

Fig. 2.5 Spiral development model

Spiral development is considered as a miniature form of the incremental model.

Limitations of Spiral Development Spiral models represent requirement elicitation as the software is being developed. Sometimes, it may lead to refactoring and changes in approach where initial structures become non-usable. Spiral development also needs huge regression testing cycles to find whether additions in the given system have affected overall system working or not.

2.8.5 PROTOTYPE DEVELOPMENT APPROACH/MODEL

Prototype development approach represents top to bottom reverse integration approach. Major problem of software development is procuring and understanding the customer requirements for the product. Prototyping is one of the solutions to help in this problem.

In prototyping, initially a prototype of the system is created—this is similar to cardboard model of a building. It helps the customer to understand what they can expect from the given set of requirements. It also helps the development team to understand the possible application's look and feel. Once the elicitation is done, the logic is build behind it to implement the elicited requirements.

Limitations of Prototype Development Though, one may get a feel of the system by looking at the prototype, one must understand that it is not the actual system but a model. The customer may get the feeling that the system is already ready and may pressurise development team to deliver it immediately. (Applications not having much graphical user interfaces are difficult to model.)

2.8.6 RAPID APPLICATION DEVELOPMENT APPROACH/MODEL

Rapid application development is not a rapid way of developing software as one may interpret from the name. It is one way to create usable software at a fast speed, and still give an opportunity to the user to understand the development and application being created.

It is a miniature form of spiral development. Development team may get very less number of requirements (let us say 5/6). They create a design, code it, test it and release it to customer. Once customer gets the delivery, he may have a better understanding of his expectations and development process by looking at

the product delivered. He may add another chunk of requirements and entire development cycle is followed. Thus, each iteration will give better understanding about a product being developed and may help in refining the requirements.

Limitations of Rapid Application Development Change in approach and refactoring are the major constraints in rapid application development. It also involves huge cycles of retesting and regression testing. Efforts of integration are huge.

2.8.7 AGILE DEVELOPMENT APPROACH/MODEL

Agile development methodologies are becoming popular due to their dynamic nature and easy adaptability to the situation. One of the surveys indicated that in case of waterfall model, many functionalities are added in requirement statement with a fear that changes in scope would not be appreciated by the development team. Some surveys show that many of the functionalities (about ¾th) developed using waterfall or iterative model are never used by the users. Agile gives complete freedom to the user to add requirements at any stage of development, and development team has to accept these changes. Agile methodologies work on small chunk of work in each iteration and release working software at the end of iteration. The main thrust of Agile methodologies is complete adaptability to user environment and continuous integration of a product. It also gives importance to delivering working software rather than achieving requirements defined in requirement specifications. Agile represents a family of development methodologies and there are many methodologies under its umbrella. Some of them are as listed below.

- Scrum
- Extreme Programming
- Feature Driven Development
- Test Driven Development

Agile works on the following principles,

- Individuals and interactions are more important than formal sign-offs for requirements, designs, etc. It concentrates more on 'Fitness for use' and what the customer needs are.
- Working software is the outcome of each milestone rather than concentrating on deliverables as defined in the project plans. Success of software product is that it is working at each stage.
- Customer collaboration is required to get usable software rather than signing various documents for approvals. Requirement clarifications, requirement elicitation, and prototyping need customer involvement.
- Responding to changes required by the customer at any moment. There may be many changes suggested by the customer as he has better knowledge about what his business needs are.

2.8.8 MAINTENANCE DEVELOPMENT APPROACH/MODEL

Major cost of the software is in its maintenance phase. Every product including software has many defects which may create problems to its users in the long term. Every technology has a life span. New technologies may offer better services and options, and may replace existing technologies. Every now and then, technological updations are required for the software as well as system to perform better and in the most cost effective way. New functionalities may be required due to changing business needs. Maintenance activities of software may be put under 4 different groups namely,

- Products where user environment is very simple, and product failure may not add to the consequences represents the lowest level of complexity. If there is any failure, it can be restored quite fast or some other arrangements can be used, and work can be continued.

2.10.3 CRITICALITY FROM DEVELOPER'S PERSPECTIVE

This classification defines the complexity of the system on the basis of development capabilities required. It may range from very complex systems to very simple systems.

- Form based software where user inputs are taken and stored in some database. As and when required, those inputs are manipulated and shown to a user on screen or in form of a report. There is not much manipulation of data and no heavy calculations/algorithms are involved.
- Algorithm based software, where huge calculations are involved and decisions are taken or prompted by the system on the basis of outcome of these calculations. Due to usage of different mathematical models, the system becomes complex and designing, developing and testing all combinations become problematic.
- Artificial intelligent systems which learn things and use them as per circumstances are very complex systems. An important consideration is that the 'learnings' acquired by the system must be stored and used when required. This makes the system very complicated.

There may be a possibility of combination of criticalities to various extends for different products at a time. A software used in aviation can affect human life, and also huge money at a time. It may not be feasible to test it in real-life scenario. It may involve some extent of artificial intelligence where system is expected to learn and use those 'learnings'. Thus, the combination increases severity of failure further.

2.11 PROBLEMATIC AREAS OF SOFTWARE DEVELOPMENT LIFE CYCLE

Let us discuss some problematic areas of software development life cycle.

2.11.1 PROBLEMS WITH REQUIREMENT PHASE

Requirement gathering and elicitation is the most important phase in software development life cycle. Many surveys indicate that the requirement phase introduces maximum defects in the product. Problems associated with requirement gathering are,

Requirements Are Not Easily Communicated Communication is a major problem in requirement statement creation, software development and implementation. Communication of requirements from customer to development team is marked by problems of listening to customer, understanding business domain, and usage of language including domain specific terms and terminologies. The types of requirements are,

Technical Requirements Technical requirements talk about platform, language, database, operating system , etc. required for the application to work. Many times, the customer may not understand the benefits of selecting a specific technology over the other options and problems of using these technically specified configurations. Selection of technology may be done as directed by the development team or as a fashion. Development organisation is mainly responsible for definition of technical requirements for software product under development on the basis of product usage. (Technical requirements also cover this type of system, whether a stand alone or client server or web application etc, tiers present in the system, processing options such as online, batch processing etc). It also talks about configuration of machines, routers, printers, operating systems, databases, etc.

Economical Requirements Economics of software system is dependent on its technical and system requirements. The technical as well as system requirements may be governed by the money that the customer is ready to put in software development, implementation and use. It is governed by cost-benefit analysis. These requirements are defined by development team along with the customer. The customer must understand the benefits and problems associated with different approaches, and select the approach on the basis of some decision-analysis process. The consequences of any specific selection may be a responsibility of the customer but development organisations must share their knowledge and experience to help and support the customer in making such a selection.

Legal Requirements There are many statutory and regulatory requirements for software product usage. For any software application, there may be some rules and regulations by government, regulatory bodies, etc. applicable to the business. There may be some rules which keep on changing as per decisions made by government, regulatory authorities, and statutory authorities from time to time. There may be numerous business rules which are defined by customers or users for doing business. Development team must understand the rules and regulations applicable for a particular product and business.

Operational Requirements Mostly operational requirements are defined by customers or users on the basis of business needs. These may be functional as well as non-functional requirements. They tell the development team, what the intended software must do/must not do when used by the normal user. Operational requirements are derived from the business requirements. This may include non-functional requirements like security, performance, user interface, etc.

System Requirements System requirements including physical/logical security requirements are defined by a customer with the help of a development team. These include requirements for hardware, machine configurations, types of backup, restoration, physical access control, etc. These requirements are defined by customer's management and it affects economics of the system. There may be some specific security requirements such as strong password, encryption, and privilege definitions, which are also declared by the customer.

Requirements Change Very Frequently Requirements are very dynamic in nature. There are many complaints by development teams that requirement change is very frequent. Many times, development teams get confused because customer requirements change continuously. '**Customer does not know what he wants**' is a very common complaint made by development teams. As the product is being built and shown to customer, lot of new ideas are suggested. Some ideas may have significant effect on cost, schedule, and effort while some other may change the architecture, basic approach, and design of software. The time gap between requirement definition and actual product delivery also plays a major role in changing requirements. Top-down approach, rapid application development, and joint application development are some of the techniques used to develop applications by accommodating changes suggested by customers.

2.11.2 GENERALLY A UNIQUE PRODUCT IS DEVELOPED EACH TIME

No two things in the world are same, though they might appear to be similar. In case of software, no two applications are same. Even in case of simple porting (desktop to client-server to web application), software application changes significantly. The same implementation done by two different developers may differ from each other. Even the same program written by the same developer at two different instances may not match

exactly. Thus a software produced may be unique for that instance. One more fact about software product maintenance is that designers find it difficult to understand original design or approach and developers find it difficult to read the code written earlier.

2.11.3 INTANGIBLE NATURE OF PRODUCT, INTELLECTUAL APPROACH THROUGHOUT DEVELOPMENT

Software products cannot be felt by normal senses. Its existence can be felt only by disc space it occupies. There are multiple options or approaches (for example, in architure or design) possible for implementation of the same set of requirements. Some may feel that one approach is better than the other for different reasons. The capabilities of individuals and organisations vary significantly in design and development, and each may have a good justification why a certain approach is selected with respect to other.

2.11.4 INSPECTION CAN BE EXHAUSTIVE/IMPOSSIBLE

While defining exhaustive inspection, one may tend to include infinite permutations and combinations of testing. Testing of complete software product is practically impossible. It may need huge money and long time to test all possibilities, and still one may not be sure that everything is covered in testing. Testing uses a sampling theory to find the defects in the product and processes used. Testing tries to find out the lacunae in development methodology and processes used.

2.11.5 EFFECT OF BAD QUALITY IS NOT KNOWN IMMEDIATELY

Any level of exhaustive testing is not capable of testing each and every algorithm, branch, condition and combination thoroughly. There are some areas which remain untested even after the application is used over extended periods. Any problem in such areas may be discovered only when the particular situation arises. The effect of this kind of problem and situation during usage may not be known beforehand while deploying the software in use.

2.11.6 QUALITY IS INBUILT IN PRODUCT

Quality of a software product cannot be improved by testing it again and again and finding and fixing the defects. It needs to be built in the product while development using good processes and methods. Any amount of testing cannot certify that a product is defect-free. Good processes and procedures can make good software. No software can be considered as defect-free even if no defect is found in the test iteration defined for it. We can only say that no defect has been discovered till that point of time using those many test cases.

2.11.7 QUALITY OBJECTIVES VARY FROM PRODUCT TO PRODUCT/ CUSTOMER TO CUSTOMER

Quality objectives define the expectations of customer/user and the acceptance level of various parameters which must be present in a given product for accepting/using it. Quality objectives are product dependent, time dependent and are mainly driven by customers or final users. There may be a possibility of trade-off between these factors. Some people define them as test objectives as they define the priority of testing. In a small computer game for kids, cost may be more important than accuracy. On the contrary, applications developed for aeronautics need to be more accurate while cost factor may not be that important. Quality objectives are defined on the basis of factors of quality which are 'must', 'should be', and 'could be' for the

application. Degree of importance changes from product to product, customer to customer, and situation to situation. Some of the quality factors are listed below.

Compliance Every system is designed in accordance with organisational and user policies, procedures and standards. If software meets these standards, it is said to be complying with the specifications. In addition to customer defined standards, there may be few standards for different domains like aviation, medical, and automotive. Software must follow these standards when it is used by particular type of people or for a particular domain. Some regulations and laws may be imposed by the regulatory and statutory bodies.

Generally, these requirements are categorised as legal requirements. These requirements may be put in non-functional requirements.

Correctness Data entered, processed and results obtained must be accurate as per requirements of customers and/or users. Definition of correctness may change from customer to customer, application to application, and time to time. Generally, accuracy refers to mathematical accuracy, but it is not a rule. For a shopkeeper, accuracy of 0.01 may be sufficient as nothing below 1 paisa is calculated while a scientist may need an accuracy of 256 digits after decimal point as rounding off errors can cause a major problem in research work.

Ease of Use Efforts required to learn, operate, prepare input for and interpret output from the system define ease of use for an application. The normal users who are expected to use the software must be comfortable while using it. Ease of use reduces training cost for the new users dramatically. If a software application can be learned without any external help, such software is considered as the best from this perspective. If there is a requirement of training or hand holding before the software can be used, people may find it inconvenient.

Ease of use is a very important factor when large number of users are expected (for example, emailing software and mobile phones), and providing them training is a difficult task.

Maintainability Efforts required to locate and fix errors in an operational system must be as less as possible to improve its ability for maintenance. There may be some possibilities of enhancements and reengineering where good maintainable software has least cost associated with such activities. Software may need maintenance activity some time or the other to improve its current level of working. Ability of software to facilitate maintenance is termed as maintainability. Good documentation, well commented code, and requirement traceability matrix improve maintainability of a product.

Portability Efforts required in transferring software from one hardware configuration and/or software system environment to another environment defines portability of a system. It may be essential to install same software in different environments and configurations as per business needs. If the efforts required are less, then the system may be considered as highly portable. On the other hand, if the system cannot be put in a different environment, it may limit the market.

Coupling Coupling talks about the efforts required in interconnecting components within an application and interconnection of system as a whole with all other applications in a production environment. Software may have to communicate with operating system, database, and other applications when a common user is working with it. Good coupling ensures better use of environment and good communication, while bad coupling creates limitation on software usage.

Performance Amount of resources required to perform stated functions define the performance of a system. For better and faster performance requirements, more and more system resources and/or optimised

design may be required. Better performance improves customer satisfaction through better experience for users. Performance attribute may cover performance, stress and volume. Details about these will be discussed in the latter chapters of this book.

Ease of Operations Effort required in integrating the system into operating environment may define ease of operations. Ease of operations also talks about the help available to a user for doing any operation on the system (for example, online help, user manuals, operations manual). 'Ease of operations' is different from 'Ease of use' where the former considers user experience while using the system, while the latter considers how fast the system working knowledge can be acquired.

Reliability Reliability means that the system will perform its intended functions correctly over an extended time. Consistent results are produced again and again, and data losses are as less as possible in a reliable system. Reliability testing may be a base of 'Build Verification Testing (BVT)' where test manager tries to analyse whether system generates consistent results again and again.

Authorisation The data is processed in accordance with the intents of the user management. The authorisation may be required for highly secured processes which deal with huge sum of money or which have classified information. Applications dealing with classified information may need authorisation as the sensitivity of information is very important.

File Integrity Integrity means that data will remain unaltered in the system and whatever goes inside the system will be reproduced back in the same way. Accepting data in correct format, storing it in the same way, processing it in a way so that data does not get altered and reproducing it again and again are covered under file integrity. Data communication within and from one system to another may also be considered under the scope of file integrity.

Audit Trail Audit trail talks about the capability of software to substantiate the processing that has occurred. Retention of evidential information about the activities done by users for further reference may be maintained.

Continuity of Processing Availability of necessary procedures, methods and backup information to recover operations, system, data, etc. when integrity of the system is lost due to problems in the system or the environment define continuity of processing. Timeliness of recovery operations must be defined by the customer and implemented by developing organisations.

Service Levels Service levels mean that the desired results must be available within the time frame acceptable to the user, accuracy of the information must be reliable, and processing completeness must be achieved. For some applications in eBusiness, service level definition may be a legal requirement. Service level may have direct relationship with performance.

Access Control The application system resources will be protected against accidental or intentional modification, destruction, misuse or disclosure by authorised as well as unauthorised people. Access control generally talks about logical access control as well as physical access control for information, assets, etc.

2.12 SOFTWARE QUALITY MANAGEMENT

Quality management approaches talk about managing quality of a product or service using systematic ways and methods of development and maintenance. It is much above achieving quality factors as defined in software requirement specifications. Quality management involves management of all inputs and processing to the processes defined so that the output from the process is as per defined quality criteria. It talks about three levels of handling problems, namely,

Correction Correction talks about the condition where defects found in the product or service are immediately sorted and fixed. This is a natural phenomenon which occurs when a tester defines any problem found during testing. Many organisations stop at fixing the defect though it may be defined as corrective action by them. Responsibility of finding and fixing defects may be given to a line function. This is mainly a quality control approach.

Corrective Actions Every defect needs an analysis to find the root causes for introduction of a defect in the system. Situation where the root cause analysis of the defects is done and actions are initiated to remove the root causes so that the same defect does not recur in future is termed as corrective action. Corrective action identification and implementation is a responsibility of operations management group. Generally, project leads are given the responsibilities of initiating corrective actions. This is a quality assurance approach where process-related problems are found and resolved to avoid recurrence of similar problems again and again.

Preventive Actions On the basis of root causes of the problems, other potential weak areas are identified. Preventive action means that there are potential weak areas where defect has not been found till that point, but there exists a probability of finding the defect. In this situation, similar scenarios are checked and actions are initiated so that other potential defects can be eliminated before they occur. Generally identification and initiation of preventive actions are a responsibility of senior management. Project managers are responsible for initiating preventive actions for the projects. This is a quality management approach where an organisation takes preventive action so that there is no defect in the first place.

Quality management is a set of planned and systematic activities which ensures that the software processes and products conform to requirements, standards and processes defined by management, customer, and regulatory authorities. The output of the process must match the expectations of the users.

2.13 WHY SOFTWARE HAS DEFECTS?

One very important question about a product is, 'Why there are defects in the product at all?'. There is no single answer to this question. After taking so much precaution of defining and implementing the processes, doing verification and validation of each artifact during SDLC, yet nobody can claim that the product is free of any defects. In case of software development and usage, there are many factors responsible for its success/failure. Few of them are,

- There are huge communication losses between different entities as requirements get converted into the actual product. Understanding of requirements is a major issue and majority of the defects can be attributed to this.
- Development people are more confident about their technical capabilities and do not consider that they can make mistakes. Sometimes self review and/or peer review does not yield any defects.

- Requirement changes are very dynamic. As the traceability matrix is not available, impact analysis of changing requirements becomes heuristic.
- Technologies are responsible for introducing few defects. There are many defects introduced due to browsers, platforms, databases, etc. People do not read and understand release notes, and consequences of failure are attributed to technologies.
- Customer may not be aware of all requirements, and the ideas develop as the product is used. Prototyping is used for clarifying requirements to overcome this problem to some extent.

2.14 PROCESSES RELATED TO SOFTWARE QUALITY

Quality environment in an organisation is established by the management. Quality management is a temple built by pillars of quality. Culture of an organisation lays the foundation for quality temple. Every organisation has different number of tiers of quality management system definition. Figure 2.6 shows a relationship between vision, mission(s), policy(ies), goal(s), objective(s) strategy(ies) & values of organisation.

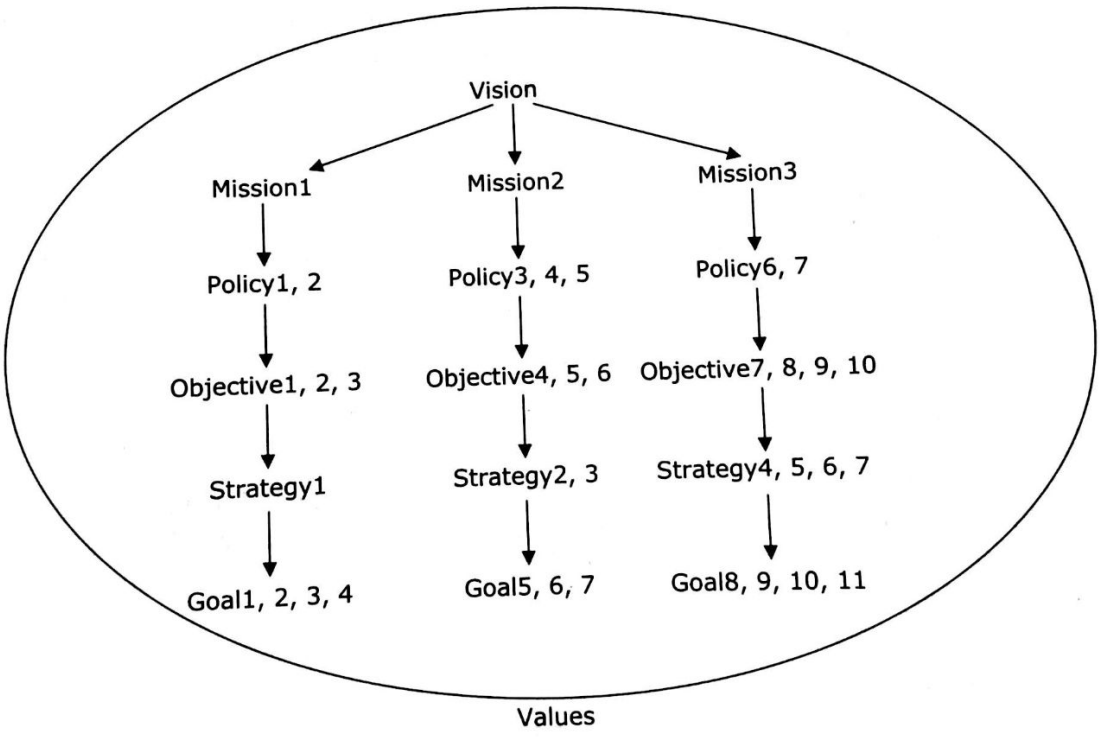

Fig. 2.6 **Relationship between Vision, Mission(s), Policy(ies), Objective(s), Strategy(ies), Goal(s) and Values**

2.14.1 VISION

The vision of an organisation is established by the policy management. Vision defines in brief about what the organisation wishes to achieve in the given time horizon. 'To become a billion-dollar company within 3 years' can be a vision for some organisations. Every organisation must have a vision statement, clearly defining the ultimate aim it wishes to achieve with respect to time span.

2.14.2 MISSION

In an organisation, there are several initiatives defined as missions which will eventually help the organisation realise its vision. Success of all these missions is essential for achieving the organisation's vision. The missions are expected to support each other to achieve the overall vision put by management. Missions may have different lifespans and completion dates.

2.14.3 POLICY

Policy statement talks about a way of doing business as defined by senior management. This statement helps employees, suppliers and customers to understand the thinking and intent of management. There may be several policies in an organisation which define a way of achieving missions. Examples of policies may be security policy, quality policy, and human resource development policy.

2.14.4 OBJECTIVES

Objectives define quantitatively what is meant by a successful mission. It defines an expectation from each mission and can be used to measure the success/failure of it. The objectives must be expressed in numerals along with the time period defined for achieving them. Every mission must have minimum one objective.

2.14.5 STRATEGY

Strategy defines the way of achieving a particular mission. It talks about the actions required to realise the mission and way of doing things. Policy is converted into actions through strategy. Strategy must have a time frame and objectives along with goals associated with it. There may be an action owner to lead the strategy.

2.14.6 GOALS

Goals define the milestones to be achieved to make the mission successful. For a mission to be declared as successful/failure at the end of the defined time frame in terms of whether the objectives are achieved or not, one needs a milestone review to understand whether the progress is in proper direction or not. Goals provide these milestone definitions.

2.14.7 VALUES

Values can be defined as the principles, or way of doing a business as perceived by the management. 'Treating customer with courtesy' can be a value for an organisation. The manner in which the organisation and management think and behave, is governed by the values it believes in.

2.15 QUALITY MANAGEMENT SYSTEM STRUCTURE

Every organisation has a different quality management structure depending upon its need and circumstances. Generic view of quality management is defined below. Figure 2.7 shows a structure of quality management system in general.

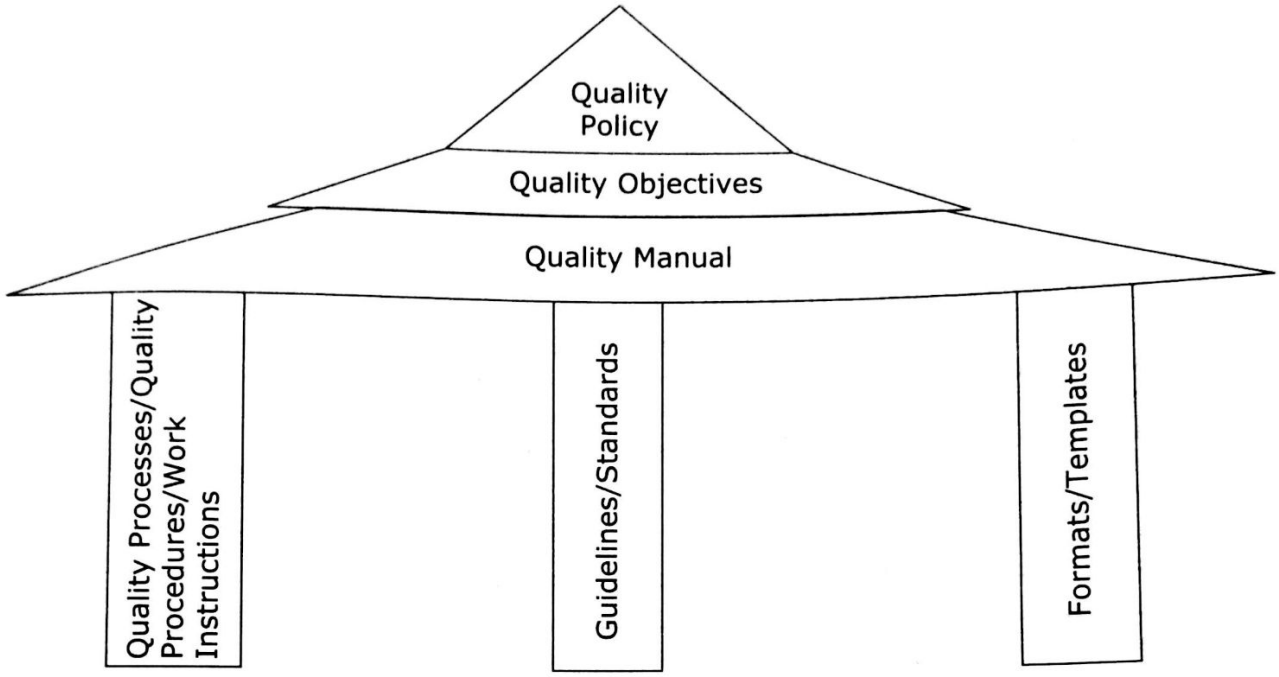

Fig. 2.7 **Quality Management System of a typical organisation**

2.15.1 1ST TIER—QUALITY POLICY

Quality policy sets the wish, intent and direction by the management about how activities will be conducted by the organisation. Since management is the strongest driving force in an organisation, its intents are most important. It is a basic framework on which the quality temple rests.

2.15.2 2ND TIER—QUALITY OBJECTIVES

Quality objectives are the measurements established by the management to define progress and achievements in a numerical way. An improvement in quality must be demonstrated by improvement in achievements of quality factors(test factors) in numerical terms as expected by the management. The achievements of these objectives must be compared with planned levels expected and results and deviations must be acted upon.

2.15.3 3RD TIER—QUALITY MANUAL

Quality manual, also termed as policy manual is established and published by the management of the organisation. It sets a framework for other process definitions, and is a foundation of quality planning at organisational level.

2.16 PILLARS OF QUALITY MANAGEMENT SYSTEM

Top part of the quality temple is build upon the foundation of following pillars.

3

FUNDAMENTALS OF SOFTWARE TESTING

OBJECTIVES

This chapter aims to provide a basic knowledge of testing. It clearly highlights the difference between 'Total Quality Management' and 'Big Bang' approaches to testing. It also defines different methodologies used in testing such as 'Black Box Testing', 'White Box Testing' and 'Gray Box Testing'. The chapter concludes with test processes including process of defining test policy, test strategy, and test plan.

3.1 INTRODUCTION

Software development activities during a life cycle have corresponding verification and validation activities at each stage of software development. Software verification involves comparing a work product with processes, standards, and guidelines. Software validation activities are associated with checking the outcome of developed product and the processes used with respect to standards and expectations of a customer. It is considered as a subset of software quality assurance activities though there is a huge difference between quality assurance and quality control. Cost of software verification as well as validation comes under appraisal cost, when one is doing it for the first time. When repeat verification/validation (such as retesting or regression testing) is done, it is defined as cost of failure. Software testing involves verification as well as validation activities such as checking the compliance of the artifacts and activities with respect to defined processes and standards, and executing the software program to ensure that it performs correctly as desired by the customer and expressed in requirement specification agreed between development team and customer. Testing involves finding the difference between actual behaviors with respect to the expected behaviors of an application. There are many stages of software testing as per software development life cycle. It begins with feasibility testing at the start of the project, followed by contract testing and requirements testing, then goes through design testing and coding testing till final acceptance testing, which is performed by customer/user.

is no possibility of software without any defect, but some level of defect may be acceptable to the customer. This approach also refers to level of confidence given to customer that application will work as expected by the user. The confidence level is determined and application is evaluated against it. Confidence-level expectations are linked with cost of testing.

3.2.5 PREVENTION-ORIENTED TESTING

Prevention-based testing is done in some highly matured organisations while for many others, the concept is still utopia. Testing is considered as a prevention activity where process problems are used to improve it so that defect-free products can be produced. Every defect found in testing is considered as process lacunae, and efforts are initiated to improve the processes of development. This reduces dependency on testing as a way to improve the quality of software. This helps in reducing the cost by producing right product at the first time.

3.3 DEFINITION OF TESTING

Testing is defined as 'execution of a work product with intent to find a defect'. The primary role of software testing is not to demonstrate the correctness of software product, but to expose hidden defects so that they can be fixed. Testing is done to protect the common users from any failure of system during usage.

This approach is based on the assumption that any amount of testing cannot show that software product is defect free. If there is no defect found during testing, it can only show that the scenario and test cases used for testing did not discover any defect. From user's point of view, it is not sufficient to demonstrate that software is doing what it is supposed to do. This is already done by system architects in system architecture design and testing, and by developers in code reviews and unit testing. Testers are involved mainly to ensure that the system is not doing what it is not supposed to do. Their work includes assurance that the system will not be exposed to any major risks of failure when a normal user is using it. Some people call this approach as negative approach of testing. This negative approach is built upon few assumptions and risks for the software being developed and tested. These assumptions and risks must be documented in the test plan while deciding test strategy or test approach.

3.3.1 WHY TESTING IS NECESSARY?

One may challenge testing activities by asking this question—'If any level of testing cannot declare that there is no defect in the product, then why is it required at all?' In normal life, we find highly qualified and experienced people involved in each stage of development from requirement gathering till acceptance of software. Finding a defect in software is sometimes considered as challenging the capabilities of these people involved in development phases. Testing is necessary due to the following reasons.

- Understanding of customer requirements may differ from person to person. One must challenge the understanding at each stage of development, and there must be some analysis of customer expectations. Approach-related problems may not be found when there is no detail analysis by another person not involved emotionally with development. Everything is considered as 'OK' unless there is an independent view of a system.
- Development people assume that whatever they have developed is as per customer requirements and will always work. But, it is imperative to create real-life scenario and undertake actual execution of a product at each level of software building (including system level) to assess whether it really works or not.

- Different entities are involved in different phases of software development. Their work may not be matching exactly with each other or with the requirement statements. Gaps between requirements, design, and coding may not be traceable unless testing is performed in relation to requirements.
- Developers may have excellent skills of coding but integration issues can be present when different units do not work together, even though they work independently. One must bring individual units together and make the final product, as some defects may be possible when the sources are developed by people sitting at different places.
- There is a possibility of blindfold and somebody has to work as the devil's representative. Every person feels that what he/she has done is perfect and there is no chance of improvement. Testers have to challenge each assumption and decision taken during development.

3.4 APPROACHES TO TESTING

There are many approaches to software testing defined by the experts in software quality and testing. The approaches may differ significantly as per customer requirements, type of the system being developed as well as management thinking about software development life cycle followed by software, type of project, type of customer, and maturity of development team. These approaches form the part of testing strategy. Few of them are discussed below.

3.4.1 BIG BANG APPROACH OF TESTING

Characteristics of 'Big bang' approach involve testing software system after development work is completed. This is also termed 'system testing' or final testing done before releasing software to the customer for acceptance testing. This testing is the last part of software development as per waterfall methodology. Big bang approach has main thrust on black box testing of software to ensure that the requirements as defined and documented in requirement specifications and design specifications are met successfully. Testing done at the end of development cycle may show the defects pertaining to any phase of development such as requirements, design, and coding. Roughly saying, the phase-wise defect origination follows the trend shown in Table 3.1.

Table 3.1 Phase-wise defect distribution

Development phases	Percentage of defects
Requirements	58
Design	35
Coding	5
Other	2

In case of big bang approach, software is tested before delivery using the executable or final product. It may not be able to detect all defects as all permutations and combinations cannot be tested in system testing due to various constraints like time. In such type of testing, one may find a cascading effect or camouflage effect, and all defects may not be detected. It may discover the failures but cannot find the problems effectively. Sometimes, defects found may not be fixed correctly as analysis and defect fixing can be a problem.

3.4.2 TOTAL QUALITY MANAGEMENT APPROACH

If there is a process definition for testing software, and these processes are optimised and capable, no (less) defects are produced and no (few) undetected defects are left in the software when it is delivered to the customer. Defect removal costs are approximately 10 times more after coding than before coding. This is a cost associated with fixing the problems belonging to requirements, design, coding, etc. If the defect is not detected earlier but found in acceptance testing or further down the line during warranty, it may be much more costly. Defect removal cost would be 100 times more during production (at user site) than before coding. This may involve deploying people at customer site, loss of goodwill, etc, which is a part of failure cost.

3.4.3 TOTAL QUALITY MANAGEMENT (TQM) AS AGAINST BIG BANG APPROACH

Figure 3.1 is a very famous cost triangle defined by TQM. If the organisation has very good processes defined which are optimised and capable, and can produce consistent results again and again, then it gives advantage in productivity and effectiveness in development. It can reduce cost of production significantly.

Stage 0 There may be a stage of maturity in an organisation where no verification/validation is required to certify the product quality. The cost involved is 'zero' or the benefits derived by investment in process definition, optimisation and deployment make quality free. There is a famous saying 'quality is free'. Theoretically, if the processes are optimised, there is no need of verification/validation as the defects are not produced at all.

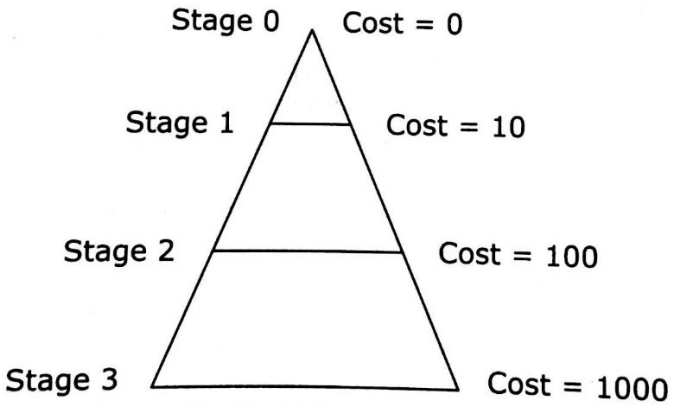

Fig. 3.1 TQM cost triangle

Stage 1 Even if some defects are produced during any stage of development in such quality environment, then an organisation may have very good verification processes which may detect the defects at the earliest possible stage and prevent defect percolation. There will be a small cost of verification and fixing, but stage contamination can be saved which helps in finding and fixing the defects very fast. This is an appraisal cost represented by '10'. This is the cost which reduces the profitability of the organisation due to scrap, rework and reverification.

Stage 2 If some defects escape the verification process, still there are capable validation processes for filtering the defects before the product goes to a customer. Cost of validation and subsequent defect fixing is much higher than verification. This cost is represented by '100'. One may have to go to the stage where defect was introduced in the product and correct all the stages from that point onward till the defect-detection point. The cost is much higher, but till that point of time the defect has not reached the customer, it may not affect customers feelings or goodwill.

Stage 3 At the bottom of the pyramid, there is the highest cost associated with the defects found by customer during production or acceptance testing. This is represented by '1000' showing that cost paid for fixing such defect is huge. There may be customer complaints, selling under concession, sending people onsite for fixing defects in front of the customer, loss of goodwill, etc. This may result into premature closure of relationship and bad advertisement by the customer.

3.4.4 TQM IN COST PERSPECTIVE

Total quality management (TQM) aims at reducing the cost of development and cost of quality through continual improvement. Often, it is termed 'Quality is free'. It means that the cost of quality must repay much more than what has been invested. TQM defines the cost incurred in development and quality into three parts as follows.

Green Money/Cost of Prevention Green money is considered as an investment by the organisation in doing quality work. It is a cost spent in definition of processes, training people, developing foundation for quality, etc. It gives return on investment, and includes all prevention-based costs. If an organisation has defined and optimised processes, trained people, and fixed guidelines and standards for doing work which are followed, then the return on such investment can be seen in terms of less inspection and testing, and higher customer satisfaction with repeat orders. This improves profitability of an organisation.

Blue Money/Cost of Appraisal Blue money is a cost incurred by the organisation during development, in the form of first-time review/testing which gets returned in future. It does not earn profit, but it is an essential part of development process to ensure the process capability. All appraisal techniques used during SDLC such as first-time verification or validation are considered as blue money. First-time testing helps in certifying that nothing has gone wrong and work product can go to the next stage. In initial phases, the cost of appraisal increases, but as the organisational process maturity increases, this cost must go down as fewer samples are required to prove the correctness of the process of making software.

Red Money/Cost of Failure Red money is a pure loss for the organisation. It involves money lost in scrap, rework, sorting, etc. It also represents loss due to various wastes produced during development life cycle, and directly reduces the profit for the organisation and the customer may not pay for it. As the investment or green money increases, failure cost must go down. All cost incurred in reinspection, retesting, and regression testing represent cost of failure.

3.4.5 CHARACTERISTICS OF BIG BANG APPROACH

Big bang approach talks about testing as the last phase of development. All the defects are found in the last phase and cost of rework can be huge.

- Testing is the last phase of development life cycle when everything is finalised. Heavy costs and efforts of testing are seen at the end of software development life cycle representing 'Big bang' approach. Most of the testing is concentrated in this phase only, as there are no previous verification or validation activities spread during the development phases.
- Big bang approach is characterised by huge rework, retesting, scrap, sorting of software components, and programs after the complete software is built. There may be special teams created to fix defects found in final testing. Big bang works only on correction, and there are no corrective and preventive actions and process improvements arising from these defects. The processes remain immature and every time, defect fixing becomes an invention of the wheel.
- Regression testing reveals many issues, as correction may not be correct and may introduce some defects in the product. There are many interdependencies between various entities while building software product, and these may get affected adversely. When some areas in the software units are touched for correction or fixing of defect, dependent components may get affected in a negative way. These may be termed 'regression defects'.
- All requirements and designs cannot be covered in testing. As the schedule of delivery is fixed and development generally lags behind the schedule, thus huge adhoc, exploratory, monkey, and random testing is done in this part of testing. Testing is done in a hurry, and many iterations of defect fixing and testing are done in the shortest possible time. Some defects may flow to customer as 'known defects' or sometimes they are not declared at all.
- The major part of software build never gets tested as coverage cannot be guaranteed in random testing. Software is generally tested by adhoc methods and intuition of testers. Generally, positive testing is done to prove that software is correct which represents second level of maturity.

Organisations following big bang approach are less matured and pay a huge cost of failure because of several retesting and regression testing along with defect-fixing cycles. Success is completely dependent on good development, testing team and type of customer. The following can be observed.

- Verification activities during software development life cycle can find out about two-third of the total number of defects found. The cost involved in fixing such defects is much less than any other way of finding the defects and fixing them. There is less stage contamination as defects are prevented from progressing from one stage of development to another.
- Validation in terms of unit testing can find out about three-fourth of the remaining defects. Defects are found in the units and fixed at that point itself so that they do not occur further down the line. It reduces the chances of defects being found in system testing. Unit testing validates an individual unit.
- Validation in terms of system testing can find out about 10% of the total number of defects. System testing must be intended to validate system-level requirements along with some aspects of design. Some people term system testing as certification testing while some people term acceptance testing as certification testing. If the exit criteria of system testing (acceptance criteria by customer) are met, system may be released to the customer.

Remaining defects (about 5–10%) go to the customer, unless the organisation makes some deliberate efforts to prevent them from leaking to production phase. Big bang approach may not be useful in preventing defects from going to the customer, as it can find only 5% of the total defects present in the product. In other terms, theoretically, to achieve the effectiveness of life-cycle testing, one may need about 18 cycles of system testing. This can prove to be a costly affair.

3.5 POPULAR DEFINITIONS OF TESTING

Let us try to define 'software testing' keeping the background of Big bang approach in mind. All definitions of testing indicate that testing is a life-cycle operation, and not the activity at the end of development phase. No definition of testing can really cover all aspects of testing. Hence, no definition is complete but indicates a part of what software testing is.

3.5.1 TRADITIONAL DEFINITION OF TESTING

There can be many definitions of testing pertaining to different instances. Few of them are as follows.

- Testing is done to establish confidence that the program does what it is supposed to do. It generally covers the functionalities and features expected in the software under testing. It covers only positive testing.
- Testing is considered as any activity aimed at evaluating an attribute or capability of a program or system with respect to user requirements. To some extent, this definition may be considered as correct, as number of defects found in testing is directly proportional to number of defects remaining in the system.
- Testing is a process of demonstrating that errors are not present in the product. This approach is used in acceptance testing where if the application meets acceptance criteria, then it must be accepted by the customer.
- Testing gives number of defects present which indirectly gives a measurement of software quality. More number of defects indicate bad software and bad processes of development.
- Testing is done to evaluate the program or system used for making software. As we consider that defects are introduced due to incapable processes, testing may be used to measure process capability to some extent.
- Testing is used to confirm that a program performs its intended functions correctly. Intended functionality may be defined from requirement specifications.

If testing is defined as a process, then it is designed to,

- Prove that the program is error free or there is no defect present
- Establish that the software performs its functions correctly and is fit for use
- Establish that all expectations of functionalities are available

Testing may not be any of these certification activities. If the goal of testing is to prove that an application works correctly, then the tester should subconsciously work towards this goal, choosing test data that would prove that the system is working correctly. The reverse would be true if the goal of testing is to locate defects so that eventually these would be corrected. Test data should be selected with an eye towards providing the test cases that are likely to cause product failure.

3.5.2 WHAT IS TESTING?

Let us try to define what is meant by testing with this background. Testing is completely guided by software requirements specifications and design specifications, and supported by test strategy and test approach depending on assumptions and risks of development, testing and usage. Testing process may include the following.

- An activity of identification of the differences between expected results and actual results produced, during execution of software application. Difference between these two results suggests that there is a possibility of defect in the process and/or work product. One must note that this may or may not be a defect.
- Process of executing a program with the intention of finding defects. It is expected that these defects may be fixed by the development team during correction, and the root causes of the defects are also found and closed during corrective actions. This can improve development process.
- Detecting specification-related errors and deviations of working application with respect to the specifications. Requirement mismatches and misinterpretation must be detected by testing.
- Establish confidence that a program does what it is supposed to do. This defines the confidence level imparted by software testing to a customer that the software will work under normal conditions. The expectation of confidence level is a function of depth and width of software testing.
- Any activity aimed at evaluating an attribute of a program or software. Acceptance testing is an activity defining whether the software has been accepted or not.
- Measurement of software quality in terms of coverage of testing (in terms of requirements, functionality, features, and number of defects found) can give information about confidence level imparted to a customer
- Process of evaluating processes used in software development. Every failure/defect indicates a process failure. This can be used to improve development processes.
- Verifying that the system satisfies its specified requirements as defined, and is fit for normal use. Requirements may be elicited with the help of the customer.
- Confirming that program performs its intended functions correctly
- Testing is the process of operating a system or component under specified conditions, observing and recording the results of such processing, and evaluating some aspect of system or component on the basis of testing
- Software testing is the process of analysing a software item to detect the difference between existing and required conditions, and to evaluate the feature of the software item.

We have previously discussed about different stakeholders and their interests in software development and testing. Let us try to analyse the expectations or views of different stakeholders about testing.

3.5.3 MANAGER'S VIEW OF SOFTWARE TESTING

The senior management from development organisation and customer organisation have the following views about testing the software product being developed.

- The product must be safe and reliable during use, and must work under normal as well as adverse conditions when it is actually used by the intended users.
- The product must exactly meet the user's requirements. These may include implied as well as defined requirements.
- The processes used for development and testing must be capable of finding defects, and must impart the required confidence to the customer.

3.5.4 TESTER'S VIEW OF SOFTWARE TESTING

Testers have different definitions about software testing, as mentioned below.

- The purpose of testing is to discover defects in the product and the process related to development and testing. This may be used to improve the product and processes used to make it.

- Testing is a process of trying to discover every conceivable fault or weakness in a work product so that they will be corrected eventually. Random testing sometimes become too imaginative, and unconceivable defects may be found.

3.5.5 CUSTOMER'S VIEW OF SOFTWARE TESTING

Customer is the person or entity who will be receiving/using the product and will be paying for it. Testers are considered as the representatives of the customer in system development.

- Testing must be able to find all possible defects in the software, alongwith related documentation so that these defects can be removed. Customer must be given a product which does not have defects (or has minimum defects).
- Testing must give a confidence that software users are protected from any unreasonable failure of a product. Mean time between failures must be very large so that failures will not occur, or will occur very rarely.
- Testing must ensure that any legal or regulatory requirements are complied during development.

Testing is an activity which is expected to reduce the risk of software's failure in production. All stakeholders have many expectations from testing. Let us try to analyse the meaning of a successful tester.

3.5.6 OBJECTIVES OF TESTING

To satisfy the definition of testing given earlier, testing must accomplish the following things.

- Find a scenario where the product does not do what it is supposed to do. This is deviation from requirement specifications
- Find a scenario where the product does things it is not supposed to do. This includes risk

The first part refers to specifications which were not satisfied by the product while the second part refers to unwanted side effects while using the product.

3.5.7 BASIC PRINCIPLES OF TESTING

The basic principles on which testing is based are given below.

- Define the expected output or result for each test case executed, to understand if expected and actual output matches or not. Mismatches may indicate possible defects. Defects may be in product or test cases or test plan.
  Developers must not test their own programs. No defects would be found in such kind of testing as approach-related defects will be difficult to find. Development teams must not test their own products. Blindfolds cannot be removed in self testing.
  Inspect the results of each test completely and carefully. It would help in root cause analysis and can be used to find weak processes. This will help in building processes rightly and improving their capability.
  Include test cases for invalid or unexpected conditions which are feasible during production. Testers need to protect the users from any unreasonable failure so that one can ensure that the system works properly.
  Test the program to see if it does what it is not supposed to do as well as what it is supposed to do.
  Avoid disposable test cases unless the program itself is disposable. Reusability of test case is important for regression. Test cases must be used repetitively so that they remain applicable. Test data may be changed in different iterations.

- Do not plan tests assuming that no errors will be found. There must be targeted number of defects for testing. Testing process must be capable of finding the targeted number of defects.
- The probability of locating more errors in any one module is directly proportional to the number of errors already found in that module.

3.5.8 SUCCESSFUL TESTERS

The definition by testers about testing talks about finding defects as the main intention of testing. Testers who find more and more number of defects are considered as successful. This gives some individuality to testing process which talks about ability of a tester to find a defect. There is a difference between executing a test case and finding the defect. This needs an ability to look for detailing, problem areas, and selection of test data accordingly.

- Testers must give confidence about the coverage of requirements and functionalities as defined in test plan.
- Testers must ensure that user risks are identified before deploying the software in production.
- Testers must conduct SWOT analysis of the software and processes used to make it. This can help in strengthening the weaker areas and the processes responsible for defects so that the same problems do not recur.

3.5.9 SUCCESSFUL TEST CASE

Testing is a big investment and justify its existence, if it catches a defect before going to the customer. Every defect caught before delivery means the probability of finding a defect by a customer is reduced. If testing does not catch any defect, it is a failure of testing and a waste for the organisation as well as customer.

Testing involved in software development life cycle starts from requirements, goes through design, coding, and testing till the application is formally accepted by user/customer.

3.6 TESTING DURING DEVELOPMENT LIFE CYCLE

Let us discuss life cycle phase and testing associated with it. This discussion is based on the consideration that development methodology follows waterfall cycle/model.

Requirement Testing Requirement testing involves mock running of future application using the requirement statements to ensure that requirements meet their acceptance criteria. This type of testing is used to evaluate whether all requirements are covered in requirement statement or not.

This type of testing is similar to building use cases from the requirement statement. If the use case can be built without making any assumption about the requirements, by referring to the requirements defined and documented in requirement specification documents, they are considered to be good. The gaps in the requirements may generate queries or assumptions which may possibly lead to risks that the application may not perform correctly. Gaps also indicate something as an implied requirement where the customer may be contacted to get the insight into business processes. It is a responsibility of business analyst to convert (as many as possible), implied requirements to expressed requirements. Target is 100%, though it is difficult to achieve.

Requirement testing differs from verification of requirements. Verification talks about review of the statement containing requirements for using some standards and guidelines, while testing talks about dummy

execution of requirements to find the consistency between them, i.e., achievement of expected results must be possible by requirements without any assumption. Verification of requirements may talk about the compliance of an output with defined standards or guidelines.

The characteristics of requirements verification or review may include the following.

- Completeness of requirement statement as per organisation standards and formats. It must cover all standards like performance and user interface expected by customer organisation.
- Clarity about what is expected by the users at each step of working while using an application. It must include the expected output by the customer. It may be in the form of error messaging, screen outputs, printer outputs, etc.
- Measurability of expected results, possibly in numerals, so that these results can be tested. Test case will have expected results which must satisfy measurement criteria defined. 'User friendliness' or 'fairly fast' are words which can create confusion about requirements.
- Testability of the scenario defined in requirement statement is must. Some requirements like application must work 24 × 7 for 10 years may not be directly testable.
- Traceability of requirements further down the development life cycle must be ensured. Requirement traceability starts at requirement phase and gets populated as one goes down the life cycle.

Theoretically, each statement in requirement document must give atleast one functional/non-functional test scenario which may result into test cases. Requirements must be prioritised as 'must', 'should be' and 'could be' requirements. The customer is an entity to confirm requirement priority.

Requirement validation must define end-to-end scenario completely so that there is no gap. It must talk about various actors, transactions involved and information transfer from one system to another.

Design Testing Design testing involves testing of high-level design (system architecture) as well as low-level design (detail design). High-level design testing covers mock running of future application with other prerequisites, as if it is being executed by the targeted user in production environment. This testing is similar to developing flow diagrams from the designs, where flow of information is tracked from start to finish. When the flow is complete, the design may be considered as good. Wherever the flow is not defined, or not clear about where it will lead to, there are defects with design which must be corrected. For low-level design, system requirements and technical requirements are mapped with the entities created in design to ensure adequacy of detail design.

Design verification talks about reviewing the design, generally by the experts who may be termed as subject-matter experts. It involves usage of standards, templates, and guidelines defined for creating these designs. Design verification ensures that designs meet their exit criteria.

- Completeness of design, in terms of covering all possible outcomes of processing and handling of various controls as defined by requirements
- Clarity of flow of data within an application and between different applications which are supposed to work together in production environment
- Testability of a design which talks about software structure and structural testing
- Traceability with requirements
- Design must cover all requirements

Code Testing Code files, Tables, Stored procedures etc are written by developers as per guidelines, standards, and detail design specifications. In reality, developers do not implement requirements directly but

they implement detail design as defined by the designer. Code testing (unit testing) is done by using stubs/drivers as required. Code review is done to ensure that code files written are,

- Readable and maintainable in future. There are adequate comments available.
- Testable in unit testing.
- Traceable with requirements and designs. Anything extra as well as anything missing can be considered as a defect.
- Testable in integration and system testing.
- Optimised to ensure better working of software. Reusability creates a lighter system.

Test Scenario and Test Case Testing Test scenarios are written by testers to address testing needs of a software application. Test cases are derived from test scenarios which are related to requirements and designs. Test scenarios can be functional as well as structural, depending upon the type of requirement and design they are addressing.

- Test scenario should be clear and complete, representing end-to-end relationship of what is going to happen and also, the possible outcomes of such processing.
- Test scenarios should cover all requirements. Test scenarios may be prioritised as per requirement priorities.
- Scenarios should be feasible so that they can be constructed during testing.
- Test cases should cover all scenarios completely.
- Test scenarios and test cases must be prioritised so that in case of less time availability, the major part of the system (where priority is higher) is tested.

3.7 REQUIREMENT TRACEABILITY MATRIX

Some quality management models and standards prescribe complete traceability of a software application from requirements through designs and code files upto test scenario, test data, test cases and test results. Requirement traceability matrix is one way of doing the complete mapping for the software. One can expect a blueprint of an entire application using requirement traceability matrix.

Typical requirement traceability matrix is as shown in Table 3.2.

Table 3.2 Requirement traceability matrix

Require-ments	High-level Design	Low-level design	Code files/ Stored Proce-dures/TBLs	Test scenario	Test cases	Test results

3.7.1 ADVANTAGES OF REQUIREMENT TRACEABILITY MATRIX

As discussed earlier, requirement traceability matrix is a blueprint of software under development. All the agencies concerned with software can use it to understand the software in a better way. It may answer questions about what is being developed and how it will be implemented. It helps in tracing if any software requirement is not implemented, or if there is a gap between requirements and design further down the line. It also helps to understand if any redundancy has been created in the application.

- Entire software development can be tracked completely through requirement traceability matrix.
- Any test-case failure can be tracked through requirements, designs, coding, etc.
- Any changes in requirements can be affected through entire work product upto test cases and vis-à-vis any test case failure can be traced back to requirements.
- The application becomes maintainable as one has complete relationship from requirement till test results available.

3.7.2 PROBLEMS WITH REQUIREMENT TRACEABILITY MATRIX

Theoretically, all softwares must have requirement traceability matrix, but in reality, most of the softwares do not have it. The reasons are numerous; some of the prominent ones are listed below.

- Number of requirements is huge. It is very difficult to create requirement traceability matrix manually. For using some tools, one needs to invest money. Also, people may need to be trained for using tools.
- There may be one-to-many, many-to-one and many-to-many relationships between various elements of traceability matrix, when we are trying to connect columns and rows of traceability matrix, and maintaining these relationships need huge efforts
- Requirements change frequently, and one needs to update the requirement traceability matrix whenever there is a change. Similarly designs, code and test cases may also change which will affect traceability matrix.
- Developing teams may not understand the importance of requirement traceability matrix, if development follows waterfall model, and during maintenance, it may be too late to create it. Incremental and iterative developments are the major challenges for maintaining traceability.
- A customer may not find value in it and may not pay for it

3.7.3 HORIZONTAL TRACEABILITY

When an application can be traced from requirement through design and coding till test scenario and test cases upto test results, it is termed as horizontal traceability. On failure of any test case, we must be able to find which requirements have not been met. Any design which does not have requirement, introduces an extra feature which may be considered as defect. Similarly, when any requirement is not traceable to design, that requirement is not implemented at all. Same thing can happen in the relationship between design and coding, coding and test scenario, and also, test scenario and test case. When any entity can't be traced in forward direction, horizontal traceability is lost.

3.7.4 BIDIRECTIONAL TRACEABILITY

One must be able to go from requirements, designs, coding, and testing to reach the test result. Reverse must also be possible, where one may start from the result and go to requirements. One must be able to go in any

direction from any point in traceability matrix. This is referred to as 'bidirectional traceability'. CMMi model mandates bidirectional traceability for all products.

3.7.5 VERTICAL TRACEABILITY

Traceability explained above is called 'horizontal traceability' as it goes in horizontal direction, either forward or backward. Traceability may exist in individual column as the requirements may have some interdependencies between them, or these may be child and parent relationships. For achieving a requirement, the other child requirement must be achieved. One requirement may have several child requirements, while some child requirements may have several parent requirements. If these requirements are traced completely, it ensures vertical traceability. Similarly design, coding, and testing may have a vertical traceability where there may be parent-child relationship and interdependence on different parts. Designs may have parent-child relationships, and coding may have 'called functions' and 'calling functions' traceability relationships.

3.7.6 RISK TRACEABILITY

Some application development organisations also add references about the risks of failure faced by the application in Failure Mode Effect Analysis (FMEA). The risks are traced to requirements and mainly with design which defines control mechanism to reduce probability or impact or improves detection ability of a risk. This helps the customer to identify which accident-prone zones are in the application and where the user is completely/partially protected from failures. It also helps in identifying various types of controls that are designed and used. Typical risk traceability matrix is as shown in Table 3.3.

Table 3.3 Risk traceability matrix

Risk	High-level Design/Control	Low-level design/Control	Code files/ Stored procedures/TBLs	Test scenario	Test cases	Test results

3.8 ESSENTIALS OF SOFTWARE TESTING

Software testing is a disciplined approach. It executes software work products and finds defects in it. The intention of software testing is to find all possible failures, so that eventually these are eliminated, and a good product is given to the customer. It intends to find all possible defects and/or identify risks which final user may face in real life while using the software. It works on the principle that no software is defect free, but less risky software is better and more acceptable to users. The tester's job is to find out defects so that they will be eventually fixed by developers before the product goes to a customer. Completion of testing must yield number of defects which can be analysed to find the weaker areas in the process of software development. No amount of testing can show that a product is defect free as nobody can test all permutations and combinations possible in the given software. Software testing is also viewed as an exercise of doing a SWOT analysis of software product where we can build the software on the basis of strengths of the process of development and testing, and overcome weakness in the processes to the maximum extent possible.

Strengths Some areas of software are very strong, and no (very less) defects are found during testing of such areas. The areas may be in terms of some modules, screens, and algorithms, or processes like requirement definition, designs, coding, and testing. This represents strong processes present in these areas supporting development of a good product. We can always rely on these processes and try to deploy them in other areas.

Weakness The areas of software where requirement compliance is on the verge of failure may represent weak areas. It may not be a failure at that moment, but it may be on the boundary condition of compliance, and if something goes wrong in production environment, it will result into defect or failure of software product. The processes in these areas represent some problems. An organisation needs to analyse such processes and define the root causes of problems leading to these possible failures. It may be attributed to some aspects in the organisation such as training, communication, etc.

Opportunity Some areas of the software which satisfy requirements as defined by the customer, or implied requirements but with enough space available for improving it further. This improvement can lead to customer delight (it must not surprise the customer). These improvements represent ability of the developing organisation to help the customer and give competitive advantage. It decides the capability of the developing organisation to provide expert advice and help to the customer for doing something better.

Threats Threats are the problems or defects with the software which result into failures. They represent the problems associated with some processes in the organisation such as requirement clarity, knowledge base and expertise. An organisation must invest in making these processes stronger. Threats clearly indicate the failure of an application, and eventually may lead to customer dissatisfaction.

3.9 WORKBENCH

Workbench is a term derived from the engineering set-up of mass production. Every workbench has a distinct identity as it takes part in the entire development life cycle. It receives something as an input from previous workbench, and gives output to the next workbench. This can be viewed as a huge conveyor belt where people are working their part while the belt is moving forward. The complete production and testing process is defined as set of interrelated activities where input of one is obtained from the output of previous activity, and output of that activity acts as an input to the next. Each activity represents a workbench. A workbench comprises some procedures defined for doing a work, and some procedures defined to check the outcome of the work done. The work may be anything during software development life cycle such as collecting the requirements, making designs, coding, testing, etc. Organisational process database refers to the methods, procedures, processes, standards, and guidelines to be followed for doing work in the workbench as well as for checking whether the processes are effective and capable of satisfying what customer is looking for in the outcome. There are standards and tools available for doing the work and checking the work in the work bench. While checking/testing a work product in a workbench, if one finds deviations between expected result and actual result, it may be considered as defect, and the work product and the process used for development needs to be reworked.

3.9.1 TESTER'S WORKBENCH

Tester's workbench is made of testing process, standards, guidelines and tools used for conducting tests, and checking whether the test processes applied are effective or not. For every workbench, there should be

a definition of entry criteria, process of doing/checking the work, and exit criteria. For testers, there must be a definition of all things that enter the testers workbench. These may be defined in a test plan. Let us discuss with an example of test-case execution as one activity represented by a workbench.

Examples of Tester's Workbench Considering a typical system testing life cycle for a product/project, the different work benches for a tester may be defined as follows. Kindly note that it is not an exhaustive list but a representative one. As one goes into finer details, there may be many more workbenches in each of these defined below.

- Workbench for creating test strategy
- Workbench for creating a test plan
- Work bench for writing test scenario
- Workbench for writing test cases
- Workbench for test execution
- Workbench for defect management
- Workbench for retesting
- Workbench for regression testing

The following is a typical workbench described for system testing execution.

Inputs to Tester's Workbench Inputs may be test scenario, test cases, work products, documentation associated with a work product, test environment, or test plan depending upon the location of workbench in life cycle. The software work product is delivered to the tester as described in delivery note supplied by development team. Delivery note must contain any known issue which tester needs to know before performing testing.

Do Process The software undergoes testing as per defined test case and test procedure. This may be guided by organisational process database defining testing process. 'Do process' must guide the normal tester while doing the process.

Check Process Evaluation of testing process to compare the achievements as defined in test objectives is done by 'check process'. Check process helps in finding whether 'do processes' have worked correctly or not.

Output Output must be available as required in form of test report and test log from the test process. Output of the tester's workbench needs to have an exit criteria definition.

Standards and Tools During testing, the tester may have to use several standards and tools. Standards may include how to install the application, which steps are to be followed while doing testing, how to capture defects, etc. There may be several tools used in testing like defect management tools, configuration management tools, regression testing tools, etc.

Rework If 'check processes' find that 'do processes' are not able to achieve the objectives defined for them, it must follow the route of rework. This is a rework of 'do process' and not of work product under testing. This ensures that all incapable processes are captured so that these can be taken for improvement.

There may be two more criteria in the work bench, viz. suspension criteria for the workbench, and resumption criteria for the workbench guided by organisational policies and standards. Suspension criteria

defines when the testing process needs to be suspended or halted, whereas resumption criteria defines when it can be restarted after such halt or suspension. If there are some major problems in inputs or standards and tools required by the workbench, 'do/check process' may be suspended. When such problems are resolved the processes may be restarted.

Testing process may be defined as a process used to verify and validate that the system structurally and functionally behaves correctly as defined by expected result. Components of testing process may include the following.

- Giving inputs (program code) to tester from previous work bench
- Performing work (execute testing) using tools and standards
- Following a process of doing and checking whether test process is capable or not
- Produce output (test results and test log) which may act as an input to the next work bench

Check process must work to ensure that results meet specifications and standards, and also that the test process is followed correctly. If no problem is found in the test process, one can release the output in terms of test results. If problems are found in test process, it may need rework. Figure 3.2 shows a schematic diagram of a workbench.

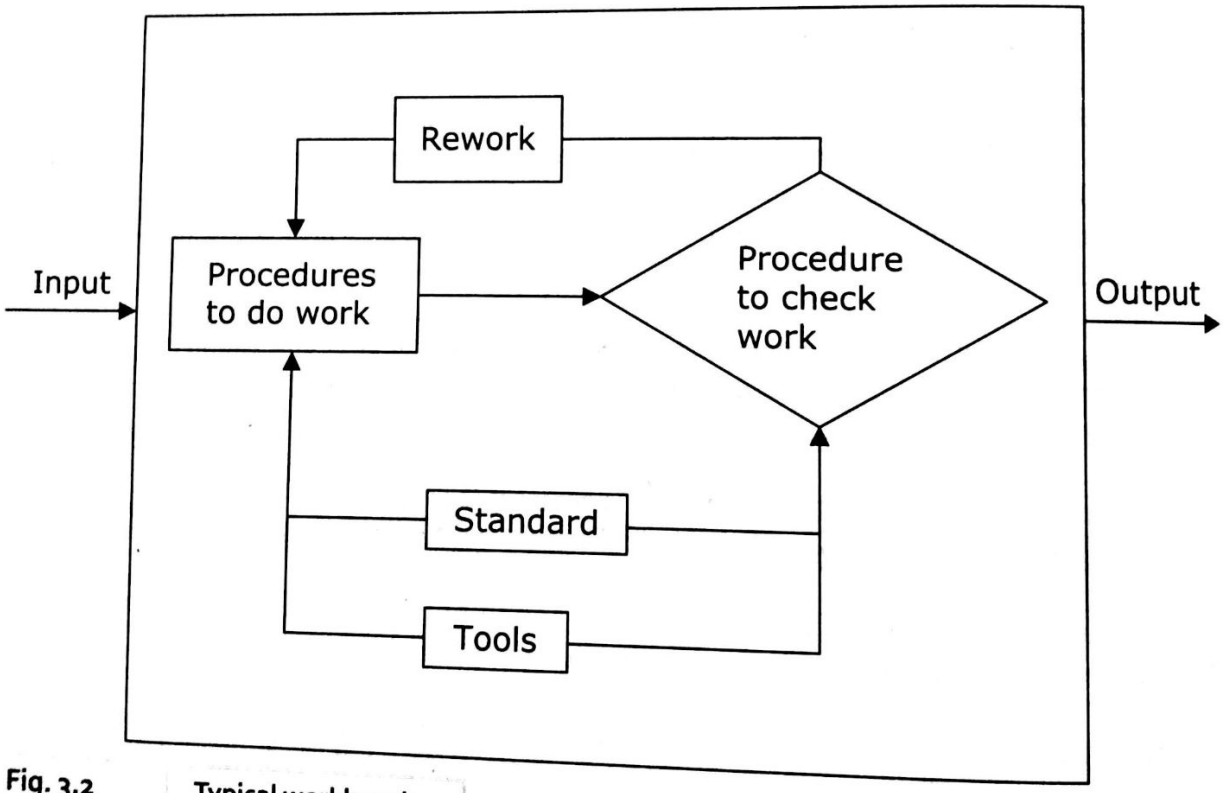

Fig. 3.2 Typical workbench

3.10 IMPORTANT FEATURES OF TESTING PROCESS

Testing is characterised by some special features, as given below.

Testing is a Destructive Process, But it is Constructive Destruction Testing involves a systematic destruction of a product with the intent to find the defects so that these would be fixed before the

product is given to the customer. While executing tests, a tester goes about testing an application using some valid/invalid inputs to find the response of software to each of these conditions. The ability of the tester to break the application can make him successful. For devising negative testing, one needs to build the scenario with destructive mentality.

Testing Needs a Sadistic Approach with a Consideration That There is a Defect A tester cannot certify that a work product is defect free. He needs to go through the software work product and hunt for defects. If the defect is not found, it directly means that there is some problem with testing process. Defect-free product does not exist. Testers are expected to identify the risks to the final users and test the product accordingly.

If the Test Does Not Detect a Defect Present in the System, it is an Unsuccessful Test Success of testing lies in its ability to find defects or threats and weaker areas of software work product and the processes supporting these areas. Testers who cannot find defects are unsuccessful testers. Testing is considered as an investment only when it reduces the probability that software may fail at customer end.

A Test That Detects a Defect is a Valuable Investment for Development As Well As Customer, it Helps in Improving a Product The root cause analysis of defects can show where the application and process can be and needs to be improved. It also helps in identifying the weaker areas of the processes used for developing software. Weaker areas are analysed to find the process lacunae and take actions on these areas and strengthen them. Thus, defect finding helps in improving processes which can result in increasing customer satisfaction. Testing with an intention to find and fix the problems in processes can be considered as an investment by customer/organisation.

Some organisations use final black box testing as an acceptance testing for the application. If no defect is found in system testing, the software is delivered to the customer. If the sole purpose of testing is to validate specifications implemented then,

- Testing is an unnecessary/unproductive activity as it does not consider invalid scenarios. No amount of testing can certify that there is no defect in the product. There must be a probability of finding the defect in testing.
- Testing is designed to compensate for ineffective software development process, it cannot give results. The defect indicates a process deficiency and it must be fixed by improving the processes. Testing cannot improve the quality of product. Defect is a symptom of something failing and one must try to fix the root cause and not only the symptom.
- If development methodology designed and implemented by a team is not correct, testing cannot compensate for it. Testing is not meant to certify the work products but to find the defects.
- Testing is a separate discipline and may get affected by as well as affects software development processes Better processes of development and testing can reduce the chances of failure of product at customer plac and subsequent customer complaints.

It is Risky to Develop Software and Not to Test it Before Delivery Not providing sufficier resources, time and support for testing activities is a common scenario across the industry. There is a beli that less-tested software means less defects, less rework, less scrap, and less-corrective actions which mea higher profits. But this may result into customer dissatisfaction as the defects will get exposed at custom site. Reducing the coverage of testing is another risk associated with software. Software testing must give desired level of confidence to users that system will not fail. Less testing reduces this confidence level a increases a probability of failure at customer site.

With High Pressure to Deliver Software As Quickly As Possible, Test Process Must Provide Maximum Value in Shortest Timeframe This approach is adopted in test strategy designing where the efficiency and effectiveness of testing is defined. Generally, test cases are categorised into installation testing, smoke testing, and sanity testing before going into further detailed testing. Test cases which represent scenarios that may occur with higher probability can help in reducing probabilities of failure in production environment. An organisation may define such test cases with probability aspect such as high, medium, and low or allocate the numbers indicating priority of execution. Probability of occurrence of a defect may not have any relationship with severity as severity talks about type of failures.

Testing is no longer an after-programming evaluation to certify that the software works, but supposed to indicate the confidence level that product will work at customer site. Testing can give the SWOT analysis of development process. Thus, testing is adjunct to software development life cycle.

Testing starts at the proposal level and ends only when software application is finally accepted by user/customer. Every stage of development and every work product must go through stages of review and formal approval to ensure that it can go to the next stage. But it is a key to ensure quality at each work product and each phase of software development life cycle.

Highest Payback Comes from Detecting Defect Early in Software Development Life Cycle and Preventing Defect Leakage/Defect Migration from One Phase to Another Defects are the problems or something wrong happening in software as well as the development process used for making software. Every defect indicates failure of the process at some place or another. It is always economical to fix the defects as and when they appear, and conduct an analysis to find the root causes of the defects rather than waiting till it hits the software product and user, again and again. It is always beneficial to do a root cause analysis and fix the problem areas at the earliest possible time. The investment in testing can be worthwhile only if it is capable of finding defects as soon as it is introduced in the work product and also can prevent any potential defect from getting introduced. Defect, if not corrected in the phase where it is introduced, leaks to the next stage and creates a larger problem. This is termed 'phase contamination'. Defect keeps on migrating from one phase to next phase, till it hits back the users at some point of time.

Organisation's Aim Must Be Defect Prevention Rather That Finding and Fixing a Defect The major misconception about testing is that it is considered as a fault-finding mission. Instead, it must be viewed as defect-prevention mission to avoid critical problems in software development process by initiating actions to prevent any recurrence of problems. Finding and fixing the problem is not a good approach as the basic cause of defect is never addressed. It needs analysis of root causes, and defect prevention mechanism—that is installed and operational—to prevent recurrence as well as removal of potential problems.

3.11 MISCONCEPTIONS ABOUT TESTING

At many places, software testing is termed 'quality assurance (QA)' activity. In reality testing is a quality control (QC) activity. There are many other misconceptions about software testing, as listed below.

Anyone Can Do Testing, and No Special Skills Are Required for Testing Many organisations have an approach that anyone can be put in testing. They give the task of testing to developers on the bench or people asking for 'light duty'. Many people involved in testing do not have any experience in testing or in the domain in which testing is done. Test planning, test case writing, and test data definition using

3.14 TEST POLICY

Test policy is generally defined by the senior management covering all aspects of testing. It decides the framework of testing and its status in overall mission of achieving customer satisfaction. For project organisations, test policy may be defined by the client while for product organisation, it is decided by the senior management.

3.15 TEST STRATEGY OR TEST APPROACH

Test strategy defines the action part of test policy. It defines the ways and means to achieve the test policy. Generally, there is a single test policy at organisation level for product organisations while test strategy may differ from product to product, customer to customer and time to time. Some of the examples of test strategy may be as follows.

- Definition of coverage like requirement coverage or functional coverage or feature coverage defined for particular product, project and customer.
- Level of testing, starting from requirements and going upto acceptance phases of the product.
- How much testing would be done manually and what can be automated?
- Number of developers to testers.

3.16 TEST PLANNING

Test planning is the first activity of test team. If one does not plan for testing, then he/she is planning for failure. Test plans are intended to plan for testing throughout software development life cycle. Test plans are defined in the framework created by test strategy and established by test policy. Test plans are made for execution which involves various stages of software testing associated with software development life cycle. Test plan should be realistic and talk about the limitations and constraints of testing. It should talk about the risks and assumptions done during testing.

Plan Testing Efforts Adequately with an Assumption That Defects Are There All software products have defects. Test planning should know the number of defects it is intending to find by executing the given test plan. Test plan should cover the number of iterations required for software testing to give adequate confidence required by customer (to show that software will be usable). Defect found in testing is an investment in terms of process improvement opportunity. Test plan is successful if intended number of defects are found.

If Defects Are Not Found, It Is Failure of Testing Activity There are many defects in software. If no (less) defects are found in testing, then it does not mean that there are no (less) defects in the product. It may mean that the test cases are not complete or adequate, or the test data is not effective in locating defects in the software product. Testing is intended to find defects. If defects are not found, the testing process may be considered as defective. Every defect found is an investment as it reduces a probability of any defect which customer may find. If more number of defects are found, it means the development process is problematic.

Successful Tester Is Not One Who Appreciates Development But One Who Finds Defect in the Product Success of testing is in finding a defect and not certifying that application or development process is good. Successful testers can find more defects with higher probabilities of occurrences

and higher severities of failure. Tester should find defects, which have a probability of affecting common users and thus contribute to a successful application.

Testing Is Not a Formality to Be Completed at the End of Development Cycle Testing is not a certifying process. It is a life-cycle activity and should not be the last part of a development life cycle before giving the application to customer/user. Acceptance testing and system testing are the integral parts of software development where the certification of application is done by the customer but complete test cycle is much more than black box testing.

Software testing includes the following.

- Verification or checking whether a right process is followed or not during development life cycle.
- Validation or checking whether a right product is made or not as per customer's need.

Some differences between verification and validation are shown in Table 3.4.

Table 3.4 Difference between Verification and Validation

Verification	Validation
Verification is an activity where we check the work products with reference to standards, guidelines, and procedures.	Validation is an activity to find whether the software achieves whatever is defined by requirements.
Verification is prevention based. It tries to check the process adherence.	Validation is detection based. It checks the product attributes.
Verification talks about a process, standard and guidelines.	Validation talks about the product.
Verification is also termed 'white box testing' or 'static testing' as the work product undergoes a review.	Validation is also termed 'black box testing' or 'dynamic testing' as work product is executed.
Verification may be based on opinion of reviewer and may change from person to person.	Validation is based on facts and is generally independent of a person.
Verification can find about 60% of the defects.	Validation can find about 30% of the defects.
Verification involves the following. • reviews • walkthroughs • inspection • audits	Validation involves all kinds of testing. • system testing • user interface testing • stress testing
Verification can give the following. • statement coverage • decision coverage • path coverage	Validation can give the following. • requirement coverage • feature coverage • functionality coverage
Some parts of software can undergo verification only, as given below. • coding guidelines • testing • commenting • declaration of variables	Some characteristics of software can be proven by validation only, as given below. • stress testing • performance testing • volume testing • security testing

3.17 TESTING PROCESS AND NUMBER OF DEFECTS FOUND IN TESTING

Testing is intended to find more number of defects. Generally, it is believed that there are fixed number of defects in a product and as testing finds more defects, chances of the customer finding the defect will reduce. Actually the scenario is reverse. As we find more and more defects in a product, there is a probability of finding some more defects. This is based on the principle that every application has defects and every test team has some efficiency of finding defects. It is governed by the test team's defect-finding ability. Let us say, the organisational statistics shows that after considerable testing and use of application by a user, number of defects found is three per KLOC; test planning must intend to find three defects per KLOC for the program under testing. The number of defects found after considerable testing will always indicate possibilities of existing number of defects. Figure 3.3 shows a relationship between number of defects found and probability of finding more defects.

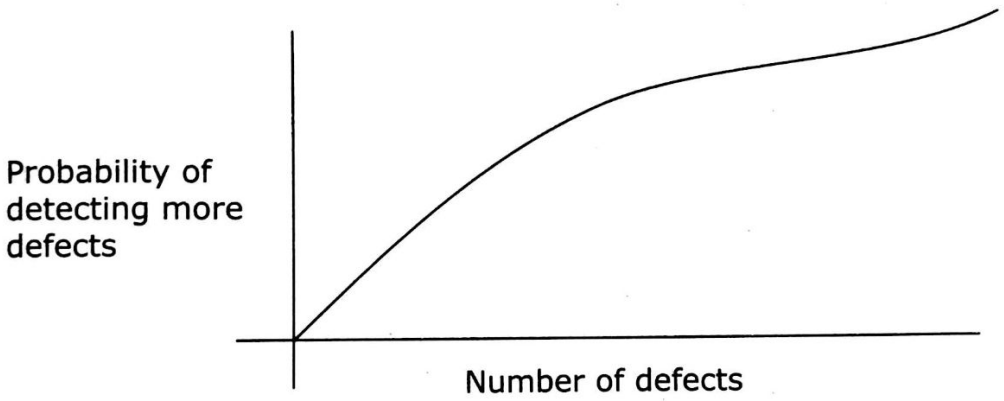

Fig. 3.3 Defect trend

3.18 TEST TEAM EFFICIENCY

Test team efficiency is a very important aspect for development team and management. If test team is very efficient in finding defects, less iterations of testing are required. On the other hand, if development team is less efficient in fixing defects, more iterations of testing and defect fixing may be required.

The test team has some level of efficiency of finding defects. Suppose the application has 100 defects and the test team has an efficiency of 90%, then it will be able to find 90 defects. Thus, if the test team finds 180 defects (considering same efficiency), it means that there are 200 defects in the software product.

Every test manager must be aware of the efficiency of a test team that he/she is working with. Often, test managers and project managers try to assess the test team efficiency at some frequency. The process may be as defined below.

Ideally, test team efficiency must be 100% but in reality, it may not be possible to have test teams with efficiency of 100%. It must be very close to 100% in order to represent a good test team. As it deviates away from 100%, the test team becomes more and more unreliable. Test team efficiency is dependent on organisation culture and may not be improved easily unless organisation makes some deliberate efforts.

There may be more thrust on integration testing rather than unit testing. Old development languages and databases sometimes make testing difficult.

Maturity of Development and Testing Processes Development and testing process maturity is an important issue in deciding the cost of testing. Theoretically, testing is not at all required if development can achieve the defined output. If development process is highly matured and can achieve the expected output, then testing may be considered as wastage. Many engineering organisations have reached the phase of 'zero' defect and 'zero' inspection. But, software application development may not have achieved a level of maturity to produce defect-free products. Some amount of testing is required to find all conceivable faults so that development team can fix them.

Many organisations have a development phase followed by one complete iteration of testing. It must find all defects so that these are eventually fixed in rework phase. Second iteration of testing must achieve the required level of confidence that application will not fail.

Testing involves all three components of cost of quality mentioned below.

3.23.2 COST OF PREVENTION IN TESTING

Cost of prevention is the cost incurred in preventing defects from entering into a system. In various phases of verification and validation, cost of prevention is distributed. Some of these are discussed below.

Cost Spent in Creation of Verification and Validation Artifacts This part of cost is spent when the project team and test team create various artifacts related to verification and validation at different phases of software development. Cost of planning includes creating test plans and quality plans so that quality can be appraised correctly. This phase may also include the time and effort spent in creation of guidelines for testing, reviews, creation of checklists, writing of test cases and test data along with test scenarios. In case of test automation, cost of test-script creation may be a part of prevention cost.

Cost Spent in Training Testers may need training on the domain under testing. They may also need training on test process and various tools used for testing. This may include organisation-level training as well as project-specific training.

Cost of prevention is calculated as follows.

- Cost spent in creating various plans related to software verification and validation. An organisation must have a baseline definition of the effort required to create plans.
- Cost spent in writing test scenarios, test cases, creating guidelines for verification and validation, and creating checklists for doing verification and validation activities. Organisation baselines must define the time and effort required for this activity.
- Training requirements may be separately assessed depending upon the competency of test teams, type of application, and level of tool usage for testing. There may not be baseline data available in this case as it may change from project to project.

3.23.3 COST OF APPRAISAL IN TESTING

Cost of appraisal includes cost spent in actually doing verification and validation activities. Generally, first-time verification/validation is considered under cost of appraisal. Test artifacts also need reviews and testing to confirm that they are correct. Checklists needed for the reviews must also be reviewed before they are used.

Irrespective of whether time and efforts are spent by developers or testers, all efforts spent on conducting reviews, walkthroughs, inspection, unit testing, integration testing, and system testing are considered under cost of appraisal.

Cost of appraisal is calculated as follows.

- Cost required for conducting first-time verification and validation activities can be assessed from the size of an application and organisation baseline data available.
- Time and effort required to conduct the given number of test cases may be derived from the productivity numbers, and number of test cases required can be derived from the size of an application.

3.23.4 COST OF FAILURE IN TESTING

Cost of failure in testing accounts for all retesting, regression testing, and rereviews conducted as the defects are found in earlier iterations. Any cost spent on and above first-time verification and validation makes a cost of failure for testing. The organisation must have a target to reduce the cost of failure continuously, as this directly affects its profitability.

Cost of failure is calculated as follows.

- On the basis of historical data available in baseline studies, one may plan number of iterations required to achieve predetermined exit criteria.
- Number of test cases per iteration and number of iterations required can give the efforts required to perform retesting and regression testing.

3.24 ESTABLISHING TESTING POLICY

Good testing is a deliberate planned effort by the organisation. It does not happen on its own, but detailed planning is required. Testing efforts need to be driven by test policy, test strategy or approach, test planning, etc. Test policy is an intent of test management about how an organisation perceives testing and customer satisfaction. It should define test objectives and test deliverables.

Test strategy or approach must define what steps are required for performing an effective testing. How the test environment will be created, what tools will be used for testing, defect capturing, defect reporting, and number of test cycles required will be a part of test strategy. It must talk about the depth and breadth of testing to ensure adequate confidence levels for users.

Test objectives define what testing will be targeting to achieve. It is better to have test objectives expressed in numbers in place of qualitative definitions. Some of the test objectives may be about code coverage, scenario coverage, and requirement coverage, whereas others may define the targeted number of defects.

Testing must be planned and implemented as per plan. Test plan should contain test objectives and methods applied for defining test scenario, test cases, and test data. It should also explain how the results will be declared and how retesting will be done. It is a general expectation that automation can solve all problems regarding testing process. One thing to be noted is that—automation can increase the speed and repeatability of testing but test planning cannot be done automatically. Rather, it needs involvement of people doing testing.

3.25 METHODS

Generally, methods applied for testing efforts are defined at organisational levels. They are generic in nature and hence, need customisation. They are customised into a test plan, and any tailoring required

to suite a specific project may be done. Management directives establish methods applied for testing. It includes what part will be tested/not tested, and how it will be tested. Which tools will be used for testing, defect-tracking mechanism, communication methods and if there is any decision of automation, how it will be undertaken—all this must be covered by these processes. Management directives are defined in test strategy.

Testing strategy may be discussed with users/customer to get their views/buy-in about testing. It may be accomplished through meetings and memorandums. User/customer must be made aware of cost of finding and fixing defects. All stakeholders for the project must be made aware that 'zero defects' is an impossible condition and acceptance criteria for the project must be defined well in advance (possibly at the time of contract). Methods of using data or inputs provided by a customer must be analysed for sufficiency and correctness.

3.26 STRUCTURED APPROACH TO TESTING

Testing that is concentrated to a single phase at the end of development cycle, just before deployment, is costly. Testing is a life cycle activity and must be a part of entire software development life cycle. If testing is done only in the last phase before delivery to customer, the results obtained may not be accurate and defect fixing may be very costly. Here, it cannot show development process problems and similar defects can be found again and again. Four components of wastes involved in this type of testing are given below.

Waste in Wrong Development Wrong specifications used for development or testing will result into a wrong product and wrong testing. Even if specifications are correct, it may be wrongly interpreted in design, code, documentation, etc. The defects not found during reviews or white box testing will be discovered only at the customer's end. This may lead to high customer dissatisfaction, huge rework, retesting, etc.

Waste in Testing to Detect Defects If testing is intended to find all defects in product, then cost of testing will be very high. Effective reviews can reduce the cost of software testing and development. If entire responsibility for software quality is left to black box testing or final system testing, then the cost of testing and cost of customer dissatisfaction may be very high.

Wastage as Wrong Specifications, Designs, Codes and Documents Must Be Replaced By Correct Specifications, Designs, Codes and Documents The cost of fixing defects may be very high in the last part of testing as there are more number of phases between defect introduction phase and defect detection phase, and defect may percolate through the development phases. Correcting the specifications, designs, codes and documents, and respective retesting/regression testing is a costly process. It can lead to schedule variance, effort variance and customer dissatisfaction. One defect fix can introduce another defect in the system.

Wastage as System Must Be Retested to Ensure That the Corrections Are Correct For every fixing of defect, there is a possibility of some other part of software getting affected in a negative manner. One needs to test software again to ensure that fixing of software has been correct, and it has not affected the other parts in a negative manner. Regression and retesting are essential when defects are found and fixed.

3.27 CATEGORIES OF DEFECT

Software defects may be categorised under different criteria. The categories of defects must be defined in the test plan. The definition may differ from organisation to organisation, project to project and customer to customer.

3.27.1 ON BASIS OF REQUIREMENT/DESIGN SPECIFICATION

- Variance from product specifications as documented in requirement specifications or design specifications represents specification related defects. These defects are responsible for 'Producer's gap'.
- Variance from user/customer expectations as business analyst/system analyst is not able to identify customer needs correctly. These variances may be in the form of implied requirements. These are responsible for 'Users gap'.

3.27.2 TYPES OF DEFECTS

- Wrongly implemented specifications are relate to the specifications as understood by developers differing significantly from what the customer wants. These may be termed 'misinterpretation of specifications'.
- Missing specifications are the specifications that are present in requirement statements but not available in the final product. The requirements are missed, as there is no requirement tracing through product development.
- Features not supported by specifications but present in the product represent something extra. This is something added by developers though these features are not supported by specifications.

3.27.3 ROOT CAUSES OF DEFECTS

- Wrong requirements given by user/customer can be a basic cause of defect. This is due to the inability of the customer to put the requirements in words, or specifying requirements which are not required.
- Business analyst/system analyst interprets customer needs wrongly can be another major cause of defect. This is due to the inability of business analyst/system analyst to elicit requirements.
- System design architect does not understand requirements correctly and hence, the architecture is wrong. This may be due to communication gap or inability of the architect in understanding the requirements.
- Incorrect program specifications, guidelines, and standards are used by respective people. If the organisation processes are not capable, then defects are introduced in the product so produced.
- Errors in coding represent lack of developer's skills in understanding design and implementing it correctly.
- Data entry error caused by the users while using a product. This can be possible when users are not protected adequately. This indicates design problems.
- Errors in testing—false call/failure to detect an existing defect in the product. The first part introduces defects in a correct product while the second part allows defects to go to the customer.
- Mistake in error correction, where defect is introduced while correcting some identified defect.

3.28 DEFECT, ERROR, OR MISTAKE IN SOFTWARE

The problems with software work product may be put under different categories on the basis of who has found it and when it has been found (as shown in Table 3.5).

3.30.7 TYPE OF DEVELOPMENT METHODOLOGY IMPACT TEST PLAN DECISIONS

Table 3.7 shows, in general, which test tactics can be used depending upon the type of development activity. This is an indicative list and may differ from situation to situation, product to product and customer to customer.

Table 3.7 Development model and testing tactics

Type	Characteristics	Test tactics
Traditional system development such as waterfall model of development	• Uses a system development methodology as defined • User knows requirements completely and these are defined • Development determines structure of application	• Test at end of each task/step/phase to ensure that exit criteria is achieved • Verify that specifications match user needs exactly • Test functions and structures as per test plan
Iterative development/prototyping development	• Complete set of requirements unknown to users and developers • Structure predefined by development • Refactoring may be done, if required	• Verify that case tools are used properly where required by testing • Test functionality first • Test other areas of product such as integration, system etc
System maintenance methodology	Modify structure if it represents a limiting factor for maintenance activities	• Test structure consistency as it may affect entire application • Works best with release methods of maintenance • Requires heavy regression testing as system is updated
Purchase/contracted software COTS	• System unknown to testers • May contain defects in hidden form • Functionality defined in user manual • Documentation may vary from software to software	• Verify that functionality matches needs of business • Test functionality as per business need • Test fitness for use into production environment

3.31 TESTING PROCESS

Testing is a process made of many milestones. Testers need to achieve them, one by one, to achieve the final goal of testing. Each milestone forms a basis on which the next stage is built. The milestones may vary from organisation to organisation and project to project. Following are few milestones commonly used by many organisations.

Defining Test Policy Test policies are defined by senior management of the organisation or test management, and may or may not form a part of the test plan. Test policy at organisation level defines the intent of test management. Test policy is dependent on the maturity of an organisation in development and test process, customer type, type of software, development methodology, etc. Test policy may be tailored at project level, depending upon the scope and historical information available from similar projects.

Defining Test Strategy Definition of test strategy includes how the test team will be organised, and how it will work to achieve test objectives, decision about coverage, automation, etc. Test strategy provides the actions to the intents defined by test policy. Test strategy helps the test team to understand the approach of testing.

Preparing Test Plan Test planning is done by test managers, test leads or senior testers, as the case may be. Test policy sets a tone, whereas test strategy adds the actions required to complete test policy. Test plan tries to answer six basic questions—What, When, Where, Why, Which and How. Test plans are for individual product/project and customer. They are derived from test strategy and give details of execution of testing activity.

Establishing Testing Objectives to Be Achieved Test objectives measure the effectiveness and efficiency of a testing process. They also define test achievements that they plan for. Test objectives are also defined from the quality objectives for the project or product, and the critical success factors for testing. Testing objectives must be 'SMART' (specific, measurable, agreed upon, realistic, and time bound).

Designing Test Scenarios and Test Cases How the test scenarios and test cases will be defined should be explained by the test strategy. A test scenario represents user scenario which acts as a framework for defining test cases. It may have actors and transactions. Similarly, there may be few scenarios arising from system requirements. Theoretically, each transaction in a test scenario eventually becomes a test case.

Writing/Reviewing Test Cases Writing and reviewing test cases along with test scenarios and updating requirement traceability matrix accordingly are the tasks done by senior testers or test leads for the project. The traceability matrix gets completed by adding the test cases and finally, the test results. One more column in the traceability matrix would be the results of execution of test cases, i.e., test results.

Defining Test Data Test data may be defined on the basis of different techniques available for the purpose. It may include boundary value analysis, error guessing, equivalence partitioning, and state transition. Test data must include valid as well as invalid set of data. It must include some special values generated from error guessing. Test data definitions may be important from testing point of view as different iterations must have different test data sets though it may be used by same test cases.

Creation of Test Bed Testing needs creation of environment for testing. It may be a real-life environment or a simulated environment using some simulators. Test bed defines some of the assumptions in a test plan which may induce certain risks of testing. It must reflect real-life situations as closely as possible. Simulators may need definition of few risks, as testing is not done in real environment.

Executing Test Cases Execution of actual test cases with the test data defined for testing the software involves applying test cases as well as test data and trying to get the actual results. It sometimes involves updating test cases or test data, if some mistakes are found in initially defined test cases or test data.

Test Result Logging results of testing in test log is the last part of the testing iteration. There may be several iterations of testing planned and executed. The defect database may be populated, if expected results are not matching with the actual results. One needs to make sure that the expected results are traceable to requirements.

Test Result Analysis Testing is a process of SWOT analysis of software under development and testing. Examining test results and analysis may lead to interpretation of software in terms of capabilities and weakness. At the end of testing, the test team must recommend the next step after testing is completed

to the project manager—whether the software is ready to go to the next stage or it needs further rework and retesting.

Performing Retesting/Regression Testing When Defects Are Resolved By Development Team When defects are given to a development team, they perform analysis and fix the defects. Retesting is done to find out whether the defects declared as fixed and verified by the development team are really fixed or not. Regression testing is done to confirm that the changed part has not affected (in a negative way) any other parts of software, which were working earlier.

Root Cause Analysis and Corrective/Preventive Actions Root cause analysis is required to initiate corrective actions. Development/test team performs post-mortem reviews to understand the weakness of development as well as test process, and initiates actions to improve them. Process improvement is the last activity after the project is closed and formally accepted by the customer. All defect prevention activities must lead to process improvements.

3.32 ATTITUDE TOWARDS TESTING (COMMON PEOPLE ISSUES)

Attitude of development team and senior management or project management towards a test team is a very important aspect to build morale of the test team. It may be initiated from test policy and may be percolated down to test strategy definition and test planning. Some of the views about test team are as follows.

- New members of development team are not accustomed to view testing as a discovery process where defects are found in the product. The defects found are taken as personal blames rather than system/process lacunae. Sometimes, people try to defend themselves, considering it as an attack on the individual.
- 'We take pride on what we developed' or 'we wish to prove that it is right' or 'it is not my fault' are very common responses. Developers may not accept the defect in the first place. If they accept the presence of a defect, then they try to put blame on somebody else. Root cause analysis is very difficult in such situations as people may attach personal ego to it.
- Conflict between developer and tester can create differences between project teams and test teams. In reality, the sole aim of development and testing must be a customer satisfaction, and defects must be considered as something which prevents achievement of this objective.

3.33 TEST METHODOLOGIES/APPROACHES

The two major disciplines in testing are given below.

Black Box Testing Black box testing is an attesting methodology where product is tested as per software specifications or requirement statement defined by business analysts/system analysts/customer. Black box testing mainly talks about the requirement specification given by customer, or intended requirements as perceived by testers. It deals with testing of an executable and is independent of platform, database, etc. This testing is with the view as if a user is testing the system.

White Box Testing White box testing is a testing methodology where software is tested for the structures. White box testing covers verification of work products as per structure, architecture, coding standards and guidelines of software. It mainly deals with the structure and design of the software product.

White box testing requires that testers must have knowledge about development processes and artifacts including various platforms, databases, etc.

There is one more methodology covering both testing methodologies at the same time.

Gray Box Testing Gray box testing talks about a combination of both approaches, viz. black box testing and white box testing at the same time. There may be various shades of black box testing as well as white box testing in this type of testing, depending upon the requirements of product. Though not a rule, gray box testing mainly concentrates on integration testing part along with unit testing.

Broadly, all other testing techniques may be put in any of these three categories, viz. black box testing, white box testing and gray box testing.

3.33.1 BLACK BOX TESTING (DOMAIN TESTING/SPECIFICATION TESTING)

Black box testing involves testing system/components considering inputs, outputs and general functionalities as defined in requirement specifications. It does not consider any internal processing by the system. Black box testing is independent of platform, database, and system to make sure that the system works as per requirements defined as well as implied ones. Actual system (production environment) is simulated, if it is difficult to create a real-life scenario in test laboratory. It does not make any assumption about technicalities of development process, platform, tools, etc. It represents user scenario and actual user interactions.

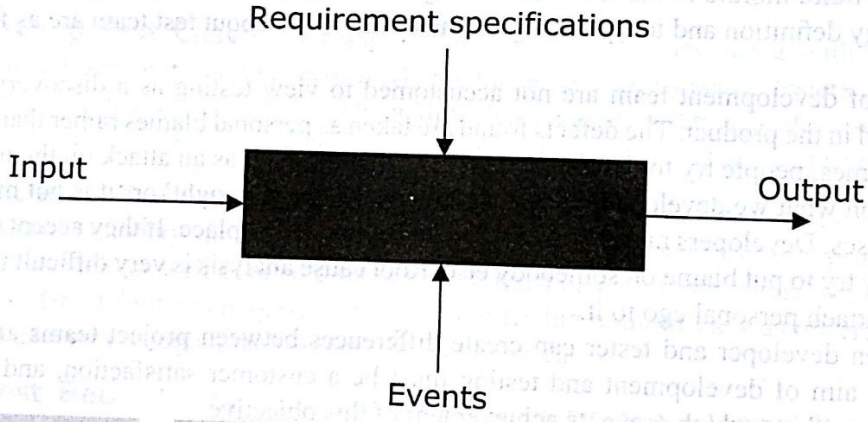

Fig. 3.8 Black box testing

Figure 3.8 shows a block box testing schematically. Black box functional testing is generally conducted for integration testing, system testing, and acceptance testing where the users/customers or the testers as representatives of the customer execute a system as if it is used by users in production environment.

Advantages of 'Black Box Testing' Black box testing is used primarily to test the behavior of an application with respect to expressed or implied requirements of the customer.

- Black box testing is the only method to prove that software does what it is supposed to do and it does not do something which can cause a problem to user/customer.
- It is the only method to show that software is living and it really works.
- Some types of testing can be done only by black box testing methodologies, for example, performance and security.

Disadvantages of 'Black Box Testing' Black box testing has many disadvantages so far as software development methodology is concerned.

- Some logical errors in coding can be missed in black box testing as black box testing efforts are driven by requirements and not by the design. It uses boundary value analysis, equivalence partitioning, and some internal structure problems can be missed.
- Some redundant testing is possible as requirements may execute the same branch of code again and again. If an application calls common functions again and again, then it will be tested so many times that it leads to redundant testing.

Test Case Designing Methodologies Black box testing methodology defines how the user is going to interact with the system without any assumption about how the system is built. As there is no view of how software is build, defining test cases is very difficult. Test cases may be defined using the user scenario called 'test scenario'. Completeness of test scenario is essential for good testing. Theoretically, each sentence in the test scenario may become a test case. Scenario contains activities in terms of transactions and actors.

Test Data Definition Black box testing is mainly driven by the test data used during testing. It may not be feasible to test all possible data which user may be using while working with an application. Some special techniques are applied for defining test data which can give adequate coverage, and also limits the number of test cases and the risk of failure of an application during use. Some of these techniques are mentioned below.

- Equivalence partitioning
- Boundary value analysis
- Cause and effect graph
- State transition testing
- Use case based testing
- Error guessing

3.33.2 WHITE BOX TESTING

White box testing is done on the basis of internal structures of software as defined by requirements, designs, coding standards, and guidelines. It starts with reviews of requirements, designs, and codes. White box testing can ensure that relationship between the requirements, designs, and codes can be interpreted. White box testing is mainly a verification technique where one can ensure that software is built correctly. Figure 3.9 shows a white box testing schematically.

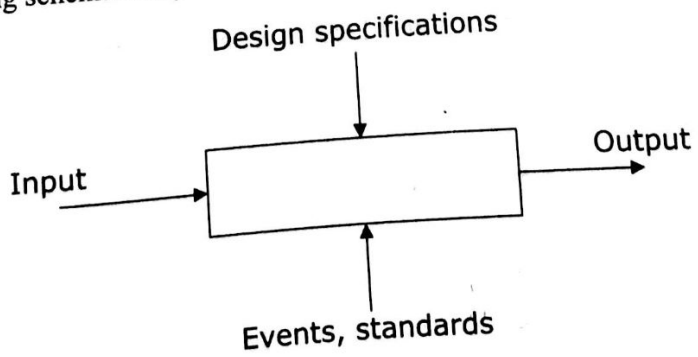

Fig. 3.9 White box testing

Advantages of 'White Box Testing' White box testing is a primary method of verification.

- Only white box testing can ensure that defined processes, procedures, and methods of development have really been followed during software testing. It can check whether the coding standards, commenting and reuse have been followed or not.
- White box testing or verification can give early warnings, if something is not done properly. It is the most cost effective way of finding defects as it helps in reducing stage contamination.
- Some characteristics of software work product can be verified only. There is no chance of validating them. For example, code complexity, commenting styles, and reuse.

Disadvantages of 'White Box Testing' White box testing being a verification technique has few shortcomings.

- It does not ensure that user requirements are met correctly. There is no execution of code, and one does not know whether it will really work or not.
- It does not establish whether decisions, conditions, paths, and statements covered during reviews are sufficient or not for the given set of requirements.
- Sometimes, white box testing is dominated by the usage of checklists. Some defects in checklists may reflect directly in the work product. One must do a thorough analysis of all defects.

Test Case Designing Test case designing is based on how test artifacts are created and used during testing. It defines how documents are written and interpreted by each person involved in software development life cycle. Some of the techniques used for white box testing are as follows.

- Statement coverage
- Decision coverage
- Condition coverage
- Path coverage
- Logic coverage

3.33.3 GRAY BOX TESTING

Gray box testing is done on the basis of internal structures of software as defined by requirements, designs, coding standards, and guidelines as well as the functional and non-functional requirement specifications. Gray box testing combines verification techniques with validation techniques where one can ensure that software is build correctly, and also works. Figure 3.10 shows a gray box testing schematically.

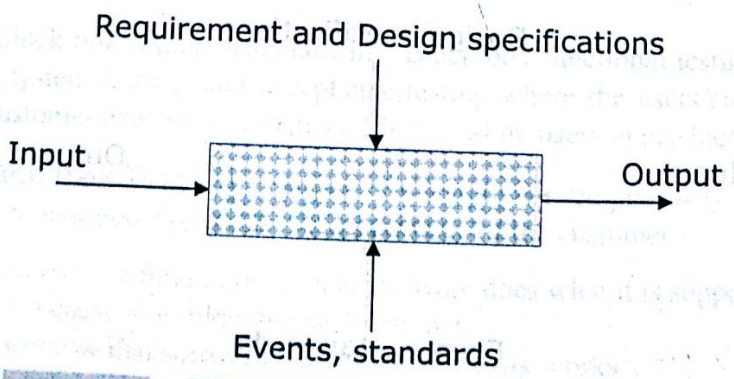

Fig. 3.10 Gray box testing

Advantages of 'Gray Box Testing'

- Gray box testing tries to combine the advantages of white box testing and Black box testing. It checks whether the work product works in a correct manner, both functionally as well as structurally.

Disadvantages of 'Gray Box Testing'

- Generally, gray box testing is conducted with some automation tools. Knowledge of such tools along with their configuration is essential for performing gray box testing.

3.34 PEOPLE CHALLENGES IN SOFTWARE TESTING

Testing is a process and must be improved continuously. People need to analyse and take actions on the shortcomings found in the process, so that they can be improved continuously. Few expectations of software process improvement needs from testers are given below.

- The tester is responsible for improving testing process to ensure better products with less number of defects going to customer, thus enhancing customer satisfaction. All defects must be found and the confidence level must be built in the process that can give customer satisfaction. Proper coverage as required by test plan must be achieved.
- Testing needs trained and skilled people who can deliver products with minimum defects to the stakeholders. Testers have to improve their skills through continuous learning.
- The tester needs a positive team attitude for creative destruction of software. Defect in the software is an opportunity to improve the product and not to blame developers. Testers must be able to pinpoint the lacunae in software development process as the defects are found.
- Testing is creative work and a challenging task. Feasible test scenarios and test cases, as well as effective ways of looking for defects are essential to improve testing effectiveness.
- Programmers and testers work together to improve the quality of software developed and delivered to customer, and the process used for software development and testing. The ultimate aim is customer satisfaction.
- Testers hunt for defects—they pursue defects not people, including developers. Every defect is considered as a process shortcoming. Defect closure needs retesting and regression testing to find whether the defect is really fixed or not, and to ensure that there is no negative impact of a defect on existing functions.
- Testing needs patience, fairness, ambition, creditability, capability, and diligence on part of testers. Every defect must be seen from the business perspective.

3.35 RAISING MANAGEMENT AWARENESS FOR TESTING

The management must be aware of the roles and responsibilities that testers are performing to achieve customer satisfaction by finding defects. If testers find defects, they can contribute in building good software by reducing probability that customer may find defects.

3.35.1 TESTER'S ROLE

While establishing a test function in an organisation, the management has some objectives to be achieved. Test team needs to understand these objectives and fulfill them.

- Calculate testing cost, effectiveness of testing and ensure that management understands the same. By doing good testing, number of customer complaints must reduce and cost of failure must go down.
- Demonstrate cost reduction and increase in effectiveness over a time span (as rework and scrap reduce over a long horizon). This can be shown by reduced customer complaints as well as less rework.
- Highlight needs and benefits of training—in test team as well as development team—on testing activities and skills, so that testers can perform better. Many developers need information about unit testing, integration testing and their role in such testing.
- Collect and distribute information on testing to all team members as well as development team/organisation which can be used for improvement.
- Get involved in test budgeting. Testing needs people, money, time, training and other resources. The organisation may have to develop budget to procure all these aspects.

3.36 SKILLS REQUIRED BY TESTER

Testing needs a disciplined approach. A tester is the person entrusted by an organisation to work as the devils agent. He/she is a person working for the client, finding the obvious defects in the processes and products. The main purpose of testing is to demonstrate that defects are present, and point towards the weaker areas in the software as well as processes used to build it, so that actions can be initiated in that direction. It must build confidence in management and customer that the application with which they will be working is usable and does not have defects. One must try to build maximum possible skills, and training is one of the effective methods to build a good testing team.

3.36.1 GENERAL SKILLS

Written and Verbal Presentation Skill Presenting test results, or discussing about an application or defects involves communication with many people. Testers are supposed to present test results and tell development team, customer and management about the present status of application and where further improvements can be done. Testers must be good in presentation skills.

Effective Listening Skill Testers need to listen to the customers voice as well as views of developers. Listening to customer as well as developer— to understand the needs and requirements correctly—is required to ensure that the scenarios and test cases can be written in a proper way. Tester's listening skills can convert testing into effective testing. Listening skills can give them complete information about the process, software application and also what the customer and management are looking for.

Facilitation Skill Facilitation of development team as well as customer is done by testers, so that defects are taken in the proper spirit. Testers must be able to tell the exact nature of a defect, how it is happening and how it will affect the users. Testers must contribute to improve development process and take part in building a better product.

Software Development, Operations and Maintenance Good knowledge of software development life cycle and software testing life cycle help testers in designing test scenarios and test cases accordingly. The defect age and cost of testing are important parameters to be controlled by testers.

Continuous Education Testers must undergo continuous education and training to build and enforce quality practices in development processes. They need to undergo training for test planning, test case definition, test data definition, methods and processes applied for testing, and reporting defects.

3.36.2 TESTING SKILLS

Concepts of Testing A tester must have complete knowledge about testing as a discipline. He/she must understand methods, processes, and concepts of testing. He/she must be capable of doing test planning, designing test scenario, writing test cases, and defining test strategy, and defining test data.

Levels of Testing Testing is a multitier activity where the application goes from one level to another after successful completion of the previous level. Testers are involved in each phase of software development right from proposal and contract, followed by requirement till acceptance testing. Testers must ensure that each phase is passed successfully.

Techniques for Validation and Verification Techniques of verification/validation must be understood and facilitated by testers to the development team, customer and management. While writing test cases, he/she needs to define test case pass/fail criteria to validate the test case and product.

Selection and Use of Testing Tools Testing involves use of various tools including automation tools, defect tracking tools, configuration management tools, and simulators. A tester must understand and use the tools effectively.

Knowledge of Testing Standards Testing standards are defined by software quality management. There can be some international standards or organisation/customer defined standards. Testers need to understand these standards, and apply them effectively so that a common understanding can be achieved.

Risk Assessment and Management Testing is risk-driven activity. Test cases and test data must be defined to minimise risks to the final users in production environment. Testing efforts must be managed to improve their effectiveness and efficiency. Managing testing involves planning, organising, directing, coordinating, and controlling testing process.

Developing Test Plan Test plan development is generally done by test managers or test leads while implementation of these plans is done by testers. Individual testers must plan for their part in overall test plan for the project.

Defining Acceptance Criteria Definition of acceptance criteria is an important milestone for testing. Generally, acceptance criteria are defined by customer well before the project starts. Testers need to define acceptance criteria for the phases and iterations of testing. There are various forms of acceptance criteria which will be discussed later. The phase-end acceptance criteria may be defined by the testers in test plan.

Checking of Testing Processes Testers follow the processes as defined in the test plan. They need to audit the testing processes to check the compliance and effectiveness, and also initiate actions if deviations are observed. Testers must contribute in testing process improvements.

Execution of Test Plan Testers are given the responsibility of executing a test plan. It includes defining test scenario, test cases, and test data, and their execution. They put test results in test log and defects in defect logging tool. Resolved defects are taken for retesting. They must perform regression testing when