CHAPTER

2

SOFTWARE QUALITY

OBJECTIVES

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

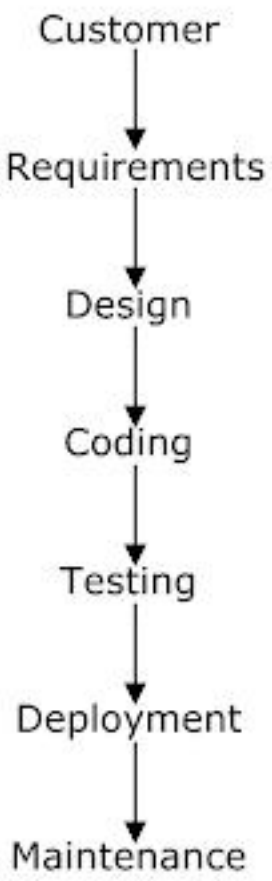

Fig. 2.2 Waterfall model

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.

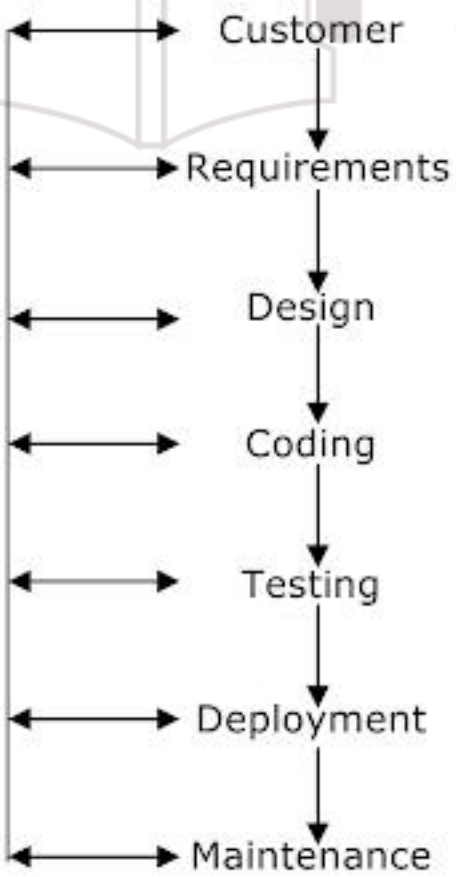

Fig. 2.3 Iterative development model

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.

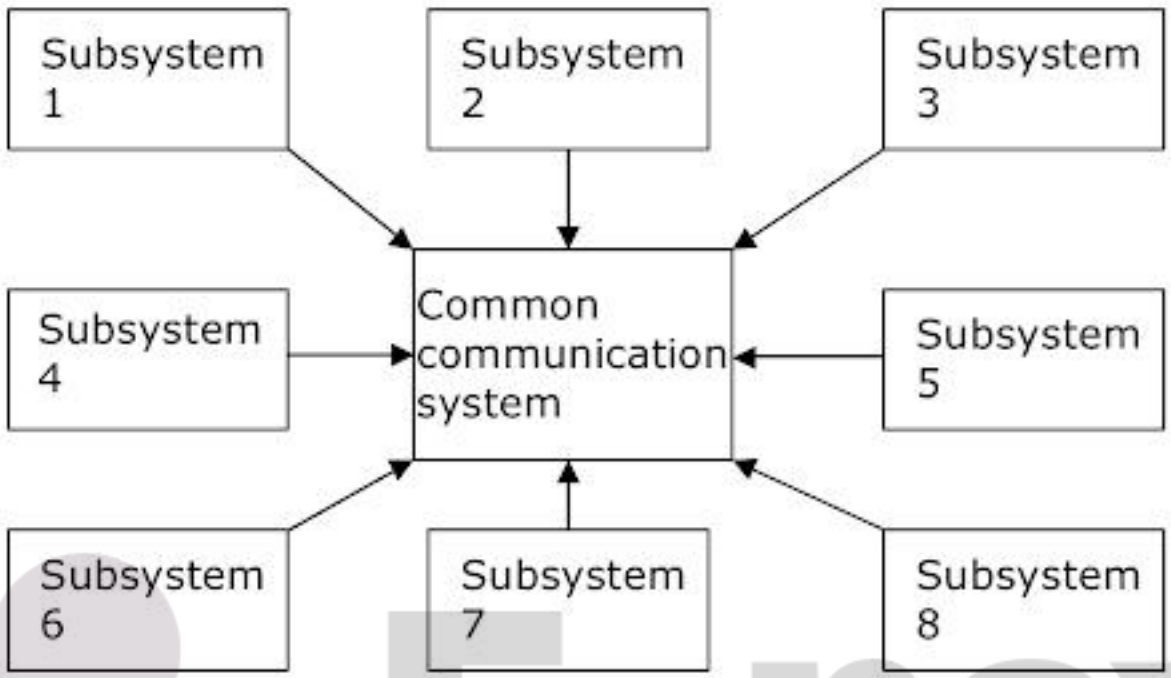

Fig. 2.4 Incremental model

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.

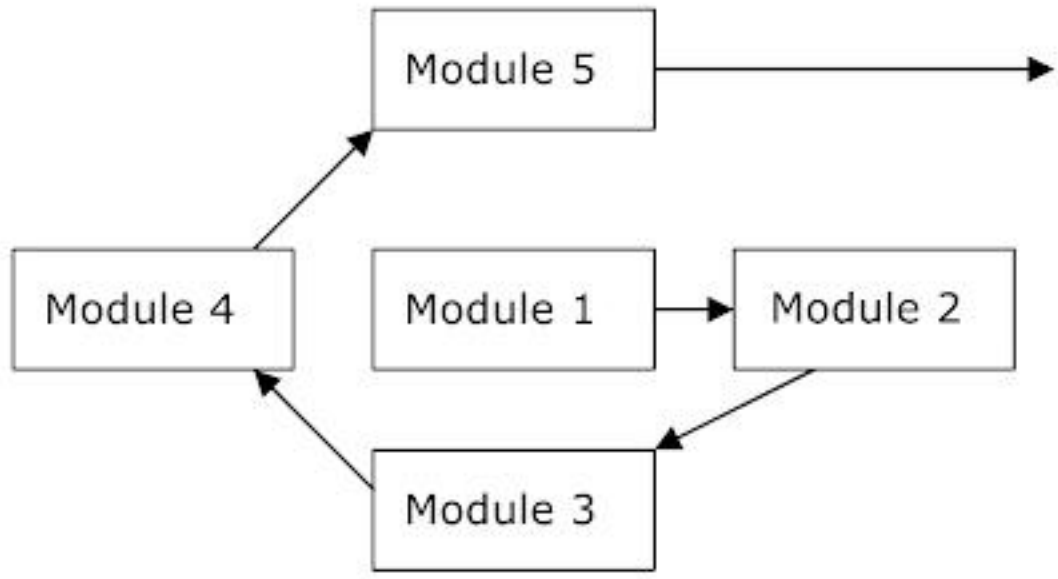

Fig. 2.5 Spiral development model

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,

- **Bug fixing** where the defects present in the given software are fixed. This may involve retesting and regression testing. During bug fixing, analysis of bug is an important consideration. There is always a possibility that while fixing a bug, new bugs may have been added in the product.
- **Enhancement** where new functionalities are added in the existing software. These functionalities may be required due to changes in the way business is done. Some functionalities may be introduced due to changes in user requirements.
- **Porting** where software is taken from older technologies to newer technologies. In porting, one is expected to port the functionalities and not the code. Whatever functionalities are available in the old technologies, all those are expected to be present in the new technology.
- **Reengineering** where there is some change in the logic or algorithm used due to changes in business environment.

2.9 TYPES OF PRODUCTS

Similar to software development methodologies, software products have some peculiarities defined as criticalities of software. Criticality of the software defines, how much important it is to a user/customer.

There are various schemes of grouping the products on the basis of criticality to the users. Few of them are listed below,

2.9.1 LIFE AFFECTING PRODUCTS

Products which directly/indirectly affect human life are considered as the most critical products in the world from user's perspective. Such products generally come under the purview of regulatory and safety requirements, in addition to normal customer requirements. The quality requirements are more stringent, and testing is very critical for such products as failures may result into loss of life or disablement of a user. This type of product may be further grouped into 5 different categories.

- Any product failure resulting into death of a person. These will be the most critical products as they can affect human life directly.
- Any product failure which may cause permanent disablement to a patient. These are second-level criticality products.
- Any product failure which may cause temporary disablement to a patient
- Any product failure which may cause minor injury and does not result into anything as defined above
- Other products which do not affect health or safety directly

Such software needs huge testing to try each and every conceivable fault in the product. It talks about very high level of confidence that application will not fail under normal and abnormal situations.

2.9.2 PRODUCT AFFECTING HUGE SUM OF MONEY

A product which has direct relationship with loss of huge sum of money is second in the list of criticality of the product. Such products may need large testing efforts and have many regulatory as well as statutory requirements. eCommerce and eBusiness softwares may be put in this category. Security, confidentiality, and accuracy are some of the important quality factors for such products.

These products also need very high confidence level and huge testing will represent the criticality. They need testing lesser than the products which directly/indirectly affect human life.

2.9.3 PRODUCTS WHICH CAN BE TESTED ONLY BY SIMULATORS

Products which cannot be tested in real-life scenario but need simulated environment for testing are third in the ranking of criticality. In this case, real life scenario is either impossible to create or may not be economically viable. Products used in aeronautics, space research, etc. may be put in this category.

Such products also need huge testing, although lesser than the earlier two types.

2.9.4 OTHER PRODUCTS

All other products which cannot be categorised in any of the above schemes may be put in this category.

Unfortunately, 'criticality' is not very easy to define. Let us consider an example of auto piloting software where we have three combinations of criticality together. It does affect the life of passengers traveling, cost of an aeroplane is huge and it cannot be tested in real environment. Thus, all the three criticalities coming together make a product most critical.

2.10 SOME OTHER SCHEMES OF CRITICALITY DEFINITIONS

There may be several other ways of classifying criticality of a product. It has direct relationship with business dependability and the extent of loss to the user organisation/person in case of failure.

2.10.1 FROM USER'S PERSPECTIVE

This classification mainly discusses dependency of a business on a system. The criticality may range from complete dependency to no/minimal dependency on the system.

- Product's failure which disrupts the entire business can be very critical from business point of view. There is no fallback arrangement available or possible in case of failure of a product which is completely dependent on the system.
- Product's failure which affects business partially as there may be some fallback arrangements is a type of criticality. Business may be affected temporarily, affecting profitability or service level but can be restored with some efforts.
- Product's failure which does not affect business at all is one of the options. If it fails, one may have another method to achieve the same result. Rearrangement may not have significant distortion of business process.

2.10.2 ANOTHER WAY OF DEFINING USER'S PERSPECTIVE

This classification considers the environment in which the product is operating. It may range from very complex user environment to very easy user environment.

- Products where user environment is very complex such as aeronautics, space research, etc. may be considered as very critical. Any failure of product can result into major problems as the environment where these products are working is not an easy environment. As the environment is very complex, system is already under stress, and failure of a product may add to the situation.
- Products where user environment is comparatively less complex (such as banks) may represent the second stage of complexity. Huge calculations may be affected, if the system collapses but there may be workaround available. People may find it inconvenient, but still the operations can be performed with some tolerable level of problems. For example, banks were running for so many years without computers and if centralised system fails, they may be able to withstand the pressure to some extent.

- 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.

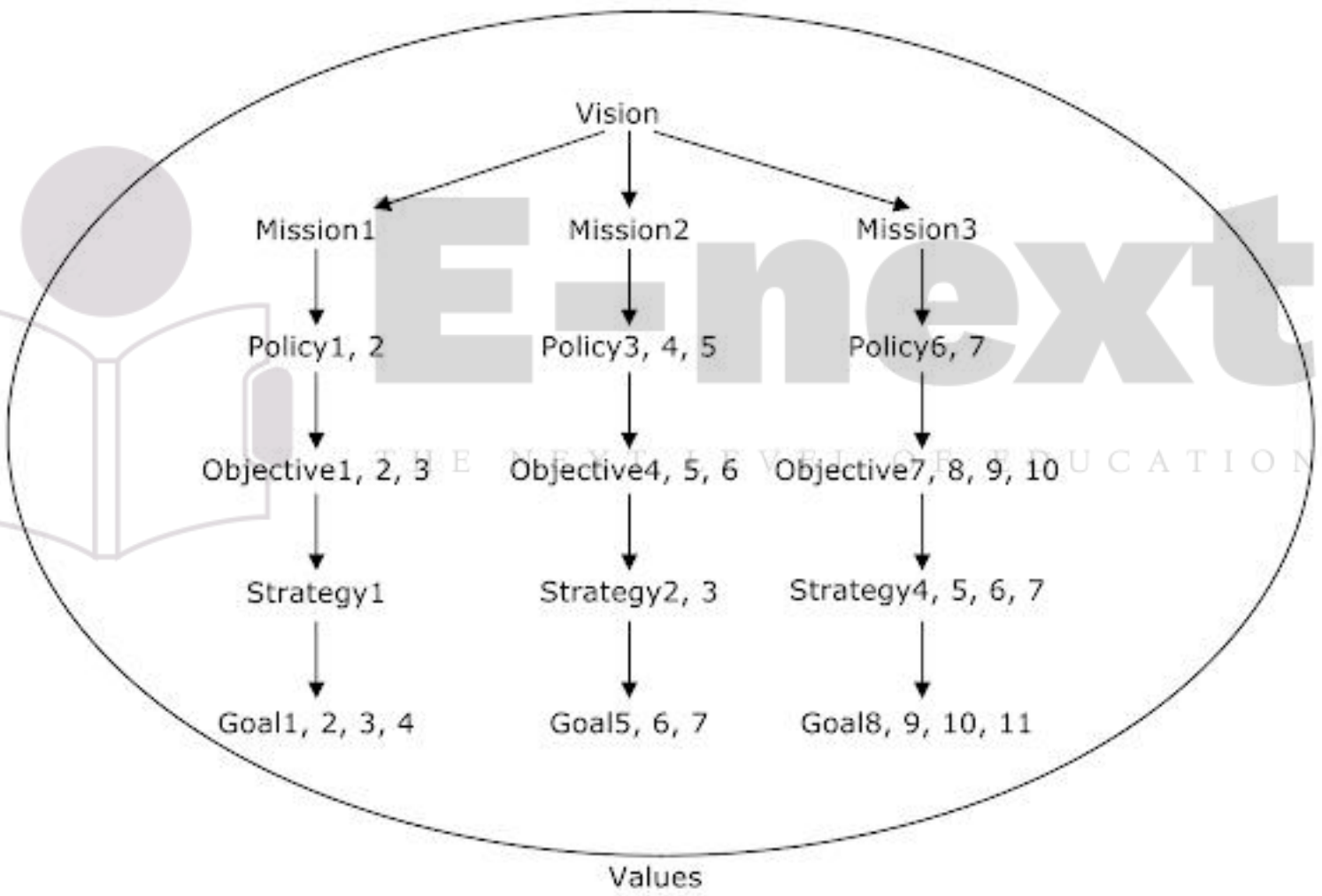

Fig. 2.6 **Relationship between Vision, Mission(s), Policy(ies), Objective(s), Strategy(ies), Goal(s) and Values**

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.

2.16.1 QUALITY PROCESSES/QUALITY PROCEDURES/WORK INSTRUCTIONS

Quality processes, quality procedures, work instructions, methods, etc. are defined at an organisation level by the functional area experts, and at project and function level by the experts in those areas separately. Organisation level processes act as an umbrella, whereas project and function level processes are in the purview of these top-level process definitions. Organisation level set of processes may differ from the process definition for different projects and functions. It is also defined as quality planning at project level. Quality procedures must be in sync with the tone established by quality manual at an organisation level.

2.16.2 GUIDELINES AND STANDARDS

Guidelines and standards are used by an organisation's project team for achieving quality goals for the products and services delivered to customers. Many a times, guidelines defined by customers are termed as standards for the project, as the project team takes the recommendations by customers as mandatory. Difference between a guideline and a standard is defined as, shown in Table 2.3.

Table 2.3 Difference between guidelines and standards

Guidelines	Standards
Guidelines are suggested ways of doing things. They are made by experts in individual fields.	Standards are mandatory ways of doing things. These are also described by experts in respective fields.
Guidelines may be overruled and there is no issue if somebody does not follow it.	Overruling of standards is a punishable offence. It may lead to nonconformance during reviews and audits.
Guidelines may or may not be written. Generally it is recommended that one must write the guidelines to capture the tacit knowledge.	Standards must be written to avoid any misunderstanding or loss of communication.
Guidelines and standards may need revision from time to time. Revisions must be done to maintain suitability over a time period.	

2.16.3 FORMATS AND TEMPLATES

Common formats and templates are used for tracking a project, function, and department information within an organisation. It creates same understanding across the board where outputs can be compared for the projects and functions. This also acts as a checklist to maintain consistency across the projects in the organisation. Formats and templates, if made compulsory, are considered as standards whereas if they are indicative or suggestive, they are considered as guidelines. Generally templates are mandatory while formats are suggestive in nature.

2.17 IMPORTANT ASPECTS OF QUALITY MANAGEMENT

Quality improvement is not an accident but a planned activity. An organisation must plan for improvement under the leadership of management and with employee participation.

2.17.1 QUALITY PLANNING AT ORGANISATION LEVEL

An organisation creates quality plan at the organisation level for achieving quality objectives, goals, its vision and missions. Quality planning includes establishing missions, policies and strategies at organisation

level along with objectives and goals to achieve the vision. It must set a framework for definition and implementation of good processes, practices, recruiting people, infrastructures, hardware and software. There should be an appraisal of quality achieved as against expected results at planned intervals, and actions must be initiated in case of any deviation.

2.17.2 QUALITY PLANNING AT PROJECT LEVEL

Projects should plan for quality at project level. These are generally strategic-level quality plans with details of responsibilities and actions. Project plan must define all aspects of quality plan at project level, and may have a relation with the organisation's quality planning. The quality objectives of the project may be inherited from organisation level objectives or may be defined separately for the project. Project level objectives must be in sync with organisation level objectives.

2.17.3 RESOURCE MANAGEMENT

An organisation should use good inputs as required by quality planning so that the output of the processes match with the organisation's business plans. It includes people, machines, materials, and methods as the basic resources. Good processes and good technology need good people to perform the work and achieve planned results.

2.17.4 WORK ENVIRONMENT

Working environment is an important input for a good product and to achieve the organisation vision and missions. A good environment can help an organisation to build on its strength while a bad environment is a roadblock to achieving objectives, and can create problems in its mission of customer satisfaction. Many organisations employ special techniques of 'Working Climate Analysis' to understand its environment, and take actions for correcting it.

Work environment has two components, viz. external environment and internal environment. External environment is mainly a physical environment while internal environment is built in the heart and brain of individuals. Good team spirit and loyalty can be major factors contributing to organisational success.

2.17.5 CUSTOMER-RELATED PROCESSES

Customer-related processes must be analysed for their capability in servicing customers and achieving customer satisfaction. Requirement analysis, designing, project processes, product delivery and other processes related to the customer must be analysed for their capabilities, and corrective/preventive actions must be initiated if those are found to be inadequate. Only capable processes can yield results in a consistent way.

2.17.6 QUALITY MANAGEMENT SYSTEM DOCUMENT AND DATA CONTROL

Many organisations define quality management system on the basis of some quality standards/models. There may be some specific customer requirements for different standards and models which may help an organisation in defining its own quality management system and following customer directives. Statistical process control and data management are essential for continuous improvement of processes.

2.17.7 VERIFICATION AND VALIDATION

Verification and validation are performed by an organisation at each level of development and for each activity. Verification includes management reviews and technical reviews (such as code review and project plan

review) whereas validation involves different kinds of testing (such as unit testing and system testing) to ensure that the work product meets the predefined acceptance criteria.

Verification Verification defines the processes used to build the product correctly. Verification is successful if the processes and procedures are followed correctly as defined by the process framework and also, they are capable of giving results. Verification cannot directly ensure that the right product has been built but checks if it has been built in the right way.

Validation Validation ensures that the right product has been built. It involves testing a software product against requirement specifications, design specifications, and customer needs. The method followed for development may or may not be correct or capable, but the final output should be as per customer requirements.

2.17.8 SOFTWARE PROJECT MANAGEMENT

Project management is a specific skill required in leaders of projects (for example, project manager). Project management involves planning, organising, staffing, directing, coordinating and controlling the project to satisfy customers by delivering the right product, on time, in the budgeted cost. Nowadays project managers have to perform different tasks like mentoring, guiding, and supporting rather than supervising people.

2.17.9 SOFTWARE CONFIGURATION MANAGEMENT

The work products are built and tested again and again. The defects found during verification and validation are corrected, and the work product undergoes further updations, integration and testing. Software configuration management involves creating work products, maintaining them, reviewing them by related stakeholders and updating them as and when required. Baselined work products are released for further development process.

2.17.10 SOFTWARE METRICS AND MEASUREMENT

New methods of project management approach stress a need for measuring the product attributes and process capabilities to achieve the quality of final deliverables to customer. Metrics and measurement programs are established by an organisation to capture metrics data/process measurement to ensure that processes are followed correctly and are capable of giving desired outputs. The lacunae found in the processes as well as products can be taken as an input to initiate corrective actions.

2.17.11 SOFTWARE QUALITY AUDITS

Audit is defined in dictionaries as an examination of accounts. Quality audits of software products and processes must be performed to analyse the quality of the products as well as processes used to make them. These can help in analysis of the situation and taking corrective/preventive actions at proper levels. Software quality audits are performed by qualified auditors at predefined levels. Audits can be categorised, as per the following:

The Auditing Agency Involved

- Internal audits are conducted by the people internal to the organisation. It is also called as first-party audits.

- Customer audits are conducted by the customer or customer representatives. It is also called as second-party audits.
- Certification audits are conducted by third-party certification bodies.

The Phase When the Audit is Conducted

- Kick-off audits are conducted at the start of the activity, say at a project start.
- Phase-end audits at the end of a phase.
- Pre-delivery audits are conducted before giving any deliverable to a customer.
- Postmortem audits are conducted at the end of the activity, say a project closure.

Subject of the Audit

- Product audits are conducted to ensure that planned arrangements for quality are achieved or not.
- Process audits are conducted to ensure that processes defined are followed or not.

2.17.12 SUBCONTRACT MANAGEMENT

Suppliers are the stakeholders for the organisation and projects. An organisation must build a strong bond of relationship with its suppliers. It must analyse the inputs from suppliers to make sure that the organisation gets proper inputs so that outputs can be managed as planned. There must be a methodology supported by values which stresses on developing a long-term relationship with the suppliers and avoids least purchase cost bids to total cost-benefit analysis. Suppliers may be supported to do statistical process control to reduce cost and delay in supply or service problems.

2.17.13 INFORMATION SECURITY MANAGEMENT

Information is one of the biggest assets of the organisation. An organisation must protect the information assets available in various databases and all information that has been developed, captured, and used. It must be able to use that information for its continuous/continual improvement. Tacit knowledge is very important from security of information. Information security is associated with three buzz words viz. confidentiality, integrity and availability.

2.17.14 MANAGEMENT REVIEW

Management must periodically review the status of different projects and functions to understand progress which in turn will help in achieving the organisation's vision. Management must plan for corrective and preventive actions if required as indicated by metrics. It must decide upon future business plans for improvements. Management reviews must be systematic and planned. Inputs, processes and outputs of management reviews must be defined.

Software quality tips

It is essential for an organisation to know whether it is achieving the target quality or not. There are some simple tips which can define the organisation's success in terms of achieving quality. Some of them are as follows,

- **Aim at Customer Satisfaction** Everything done by an organisation is for achieving customer satisfaction. Management must devise a process of collecting customer feedback and periodically measure and monitor customer satisfaction. It must initiate actions where the customer feedback is negative.

- **Have Measurable Objectives** Measurable objectives are essential to track the progress made by the organisation. Qualitative objectives may or may not be sufficient to ensure its achievement as the organisation achieves maturity, and the organisation should try to put quantitative objectives, atleast for critical processes. Organisations must have a definition of goals along with objectives for continuous measurements.

- **Understand Requirements Accurately** Understanding and defining customer requirements is the most challenging work for business analyst, system analyst and management. It needs to ensure that a developer must understand the requirements correctly and interpret the requirements into correct product. Requirement losses in terms of misinterpretation must be reduced. Implied requirements must be converted into defined requirements.

- **Implement P–D–C-A Cycle in Each Phase** Plan–Do–Check-Act cycle of continual (continuous) improvement must be followed to ensure improvement in product and process quality. An organisation must plan for future, do the things as planned, check the actual results with the planned ones and take actions on deviations.

- **Detect and Remove Defects as Early as Possible, Prevention is Better Than Cure** In software development, longer the defect remains in a system, more costly and more difficult it is to remove it. It would be always advantageous to detect the defect through review and testing process as early as possible. All defects must lead to process improvements so that defect recurrence is avoided.

- **Systematic Change Control and Version Control** Any change in work product must go through the stages of draft, review, approve and baseline. Version control and labeling is used effectively during development process to identify work products. Many tools are available for managing changes, though it can also be done manually. Configuration management is very important to give the right product to the customer.

- **Follow Easy to Use Standards/Conventions for Naming, Commenting, Coding and Documentation** An organisation must define standards and guidelines which are very useful for normal developers and testers. Common standards and guidelines show the best way to do things. It spreads common understanding across the teams and reduces the chances of misinterpretation. People do not have to invent the wheel again and again but can use the experience of others by referring to these guidelines and standards. They must be very simple to understand and use.

- **Start with Compiling and Analysing Simple Metrics** An organisation must define simple metrics at the start which are useful for planning improvements and tracking them. The main purpose of metrics is to define improvement actions needed and measuring how much has been achieved.

Summary

This chapter is based upon the foundation of the earlier chapter where we have seen quality perspectives. In this chapter, we have studied different constraints faced while building and testing a software product. It tries to link the relationship between quality improvement and productivity improvement. We have seen the cultural difference between quality conscious organisations 'Q' and less quality conscious organisations 'q'.

This chapter elucidates various development models such as waterfall, iterative, incremental, and prototyping. New development methodologies like agile have also been introduced. We have also seen the criticality definitions of different systems from the perspective of different stakeholders and how a tester must understand these system criticalities before devising testing.

We have learnt different types of requirements and problems faced while defining these requirements. We have listed all the quality factors (test factors) applicable for a system and how it affects testing.

Finally, we have dealt with quality management system as a generic model and seen that prevention is required rather than finding and fixing the defects.

1) What are the constraints of software requirement specifications?
2) Explain relationship between quality and productivity.
3) Explain the concept of 'q' organisations and 'Q' organisations
4) Explain different development models.
5) How products are classified depending upon their criticality?
6) What are different types of requirements?
7) What problems are posed by the requirement stage?
8) What are the characteristics of good requirements?
9) Explain difference between expressed and implied requirements.
10) Explain difference between present and future requirements.
11) Explain difference between generic and specific requirements.
12) List and explain quality objectives (test objectives) applicable to software development and usage.
13) Explain generic quality management system structure for an organisation.