Software Engineering with Project Management

Chapter 1 : The Software Process

Q. 1 What is software?

Ans. :

Definition

Computer software, or simply software, is a generic term that refers to a collection of data or computer instructions that tell the computer how to work, in contrast to the physical hardware from which the system is built, that actually performs the work.

In computer science and software engineering, computer software is; all the information processed by computer systems, programs and data.

Computer software includes computer programs, libraries and related non-executable data, such as online documentation or digital media.

Fig. 1.1 shows how the user interacts with application software on a typical desktop computer. The application software layer interfaces with the operating system, which in turn communicates with the hardware. The arrows indicate information flow.

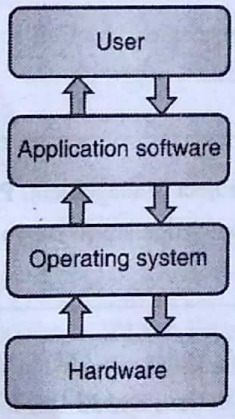


Fig. 1.1 : User interaction with application

Q. 2 Explain changing nature of software.

Ans. : Changing nature of software

Now-a-days the software is not only remains a product, it also becomes an interface for delivering a product.

As a product, software is able to deliver the computing potential which is basically embodied by computer hardware or precisely we can say, by a network of computers which one can access through local hardware.

Software is considered as an information transformer or producer irrespective of whether it resides within a smart phone or functions inside a mainframe computer. Just like any transport vehicle, software can work as the basis for the control of the computer (operating systems), the transport of information (networks) and the generation and management of other programs (such as various software tools). Information delivery is the most important aspect of software. To make any data more useful in a local context, software can transform the data. It also helps in improving the competitiveness of a business by managing the business information.

A gateway is provided by software to worldwide information networks (e.g., the Internet), and offer the way to gather information in several different forms. Over the last half-century, significant changes are occurred in the role of computer software. Now-a-days the software becomes more sophisticated as well as complex because of various aspects such as spectacular improvements in hardware performance, tremendous up-gradations in computing architectures, huge enhancements in memory and storage capacity, and extensive diversity of exotic input and output options. Sophistication as well as complexity leads to amazing results on success of systems, but they may also lead to huge problems for those who want to develop complex systems.

In the economies of the industrialized world, the big software industry is considered as a dominant factor. In an earlier era, software was developed individually. Now-a-days an entire team works for it in which every person is responsible for one part of the technology which is necessary to deliver a complex application.

Q. 3 Define software.

Ans. :

Software can be defined in different ways :

Definitions

1. **Software** is set of instructions (computer programs) that when executed provide desired features, function, and performance.
2. **Software** is data structures that enable the programs to adequately manipulate information.

Scanned by CamScanner

3. **Software** is descriptive information in both hardcopy and virtual forms that describes the operation and use of the programs.

Q. 4 Describe the characteristics of software.

Ans. : Characteristics of software

1. Software is developed or engineered; it is not manufactured in the classical sense

Even though there are some similarities found in software development and hardware manufacturing, both the activities are considered as fundamentally different. In both of the activities, good design is base for high quality, but in the process of manufacturing there may be quality problems in hardware which may not present in case of software. In both of the activities there is prominent dependency on people but there is vast difference in the relationship between people and work accomplished.

The main aim of both the activities is construction of a "product", but the perspectives are not same. Software costs are concentrated in engineering which indicates that it is difficult to manage software projects similar to manufacturing projects.

2. Software doesn't "wear out"

In case of hardware, the failure rates are high as compared to software early in its life. Such failures are mainly concerned with design or manufacturing defects. It is possible to correct the defects and drop the failure rate to a steady-state level for some time period. However after some time period, there is again increase in the failure rate as there are cumulative effects of dust, vibration, mishandling, tremendous high or low temperature, and number of other environmental maladies on components of hardware. In simple words, the hardware begins to *wear out.*

Software does not have any risks of such environmental maladies which lead to wear out of hardware. In early life of software there may be high failure rates because of undiscovered defects. However, these can be corrected. Now it is clear that there is no possibility of software wear out but it does deteriorate. In it life change is an integral part of software. Whenever any change is made, there is possibility of introduction of errors which increase the failure rate. Before the software gets consistent state by corrections, any new change get introduced which again increases the failure rate.

Gradually, the lowest failure rate level starts to increase which indicates that the software is deteriorating due to change. There is one important difference in hardware and software regarding wear aspect is that in case of hardware, a failed component can be replaced by spare parts. This facility is not available in software as there are no such spare parts. All the software failures point out that there is an error in design or in the method by which the respective design was transformed into machine executable code.

Hence the tasks of software maintenance which handles requests for change has significantly more complexity as compared to hardware maintenance.

3. Custom built software

Component reusability is an important aspect in software industry. It is responsibility of software engineer to design and implement a software component in such a way that it should be reused easily in many different programs. Latest reusable components summarize both data as well as the processing which is applied to the data, which helps the software engineer to develop new applications from existing components. For example, reusable components are used in the development of modern interactive user interfaces that enable the generation of graphics windows, pull-down menus, as well as wide range of interaction mechanisms.

4. Efficiency

Software is said to be efficient if it uses the available resources in the most efficient manner and produce desired result in timely manner.

5. Maintainability

If customer requirements changes, programmers need to modify the software to fulfill those requirements. Software engineering provides the ability to maintain the software through modifying the software rather than changing whole product like hardware.

6. Dependability

Dependability is the ability of the software that provides services which can be trusted by users. Dependability provides services to fulfill the customer's requirements, so that it is helpful to achieve trust of customers on the system.

Q. 5 Explain the term 'Software Engineering'.

Ans. :

Software Engineering combines two concepts, Software and Engineering.

Software

Software is set of executed programs related to specific functionality. Set of executable instructions is called as a program.

Software is more than just a program code; it also includes data structures which allow programs to manipulate information, and documentation related to software product which defines the functionality and guidance for the user to use this software. Software engineers design and build the software product.

Engineering

Engineering is an application of knowledge and well defined principles to develop products.

Scanned by CamScanner

Software Engineering

Definition 1

Software Engineering is the method of applying scientific and technological knowledge, procedures, and rules to design, develop, and maintain the software product.

OR

Definition 2

Applying technological, scientific, and administrative approach to designing, developing, testing and maintaining the software product in order to meet customers' requirements with best quality of product referred as software engineering.

Q. 6 Explain Layered approach of software technology May 17

Ans. :

Layered approach of software technology

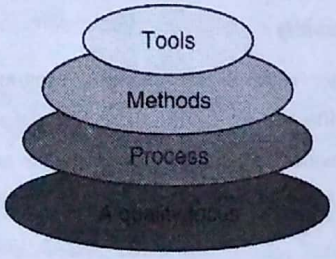


Fig. 1.2 : Layered Approach of software Technology

This approach is divided into 4 layers :

1. **A quality Process**
   (a) Any engineering approach must rest on the quality.
   (b) The most important aspect in software Engineering is Quality Focus.
2. **Process**
   (a) Foundation for SE is the Process Layer.
   (b) SE process is the GLUE that holds all the technology layers together and enables the timely development of computer software.
   (c) It forms the base for management control of software project.
3. **Methods**
   (a) SE methods provide the "Technical Questions" for building Software.
   (b) Methods contain a broad array of tasks that include communication requirement analysis, design modeling, program construction testing and support.
4. **Tools**
   (a) SE tools provide automated or semi-automated support for the "Process" and the "Methods".
   (b) Tools are integrated so that information created by one tool can be used by another

Q. 7 Explain in brief Software Process Framework.

Ans. :

Software process framework

A software engineering process is the model selected for managing the creation of software from customer initiation to the release of the finished product.

The software life cycle follows the sequential flow in order to develop software.

(i) Requirement gathering and analysis

In this phase stakeholders communicate with customer and system users to gather the business requirements like who will use the system? How user will interact with the system? What should be the system input and output? etc.

Depending on these requirements, Requirement Specification Document (SRS) is prepared.

(ii) Design

This phase makes use of output of requirement gathering phase i.e. requirement specification document as an input.

Based on requirement specifications software design is prepared. Output of this phase is design specification which specifies hardware and software requirements of system. Data Flow Diagram, flowcharts etc. are included by design specification document. Design specification acts as an input for implementation phase.

(iii) Implementation phase

Implementation phase of development process decomposes the system work into various smaller parts called modules. These modules are assigned to development team members and actual coding is started. This is longest phase of system development Output of this phase is actual code using specific programming language.

(iv) Testing

Testing phase involves testing of actual code of system against requirements of user in order to ensure that system will satisfy all needs of users. Various testing strategies used are unit testing, system testing, integration testing etc. Output of testing phase is corrected and modified software code.

Scanned by CamScanner

(v) Deployment

In this phase the system is deployed at users' site for their use. In this phase customer uses the software and give feedback to development team for any changes or modifications in system if any.

(vi) Maintenance

Once the software is deployed actual problems from user's perspective will come up. It is responsibility of development team to solve user's problems time to time. This process is called as maintenance of system. In this phase some additional functionality may need to add in the application as per user's requirement.

Q. 8 What are the different levels of Capability Maturity Model (CMM) ?

Dec. 15, May 16, Dec. 16, Dec 17, May 18

Ans. : Different levels of capability maturity model

The Capability Maturity Model (CMM) invented by Software Engineering Institute (SEI) states a growing sequence of levels of a software development organization. The higher the level, the better the software development procedure so reaching every level is a cost consuming and time consuming procedure. The Capability Maturity Model (CMM) is a mechanism used to create and refine software creation procedure of organization. The CMM is same as ISO 9001, one of the ISO 9000 sequences of standards determined by the International Organization for Standardization (ISO).

The ISO 9000 standards state an efficient quality system for manufacturing and service industries; ISO 9001 applied particularly to software development and maintenance. The major variation among the two systems present in their uses : ISO 9001 states a lowest acceptable quality level for software procedures, while the CMM use a framework for uninterrupted procedure improvement and is more explicit than the ISO standard in defining use of employed to that end.

Following are levels of CMM :

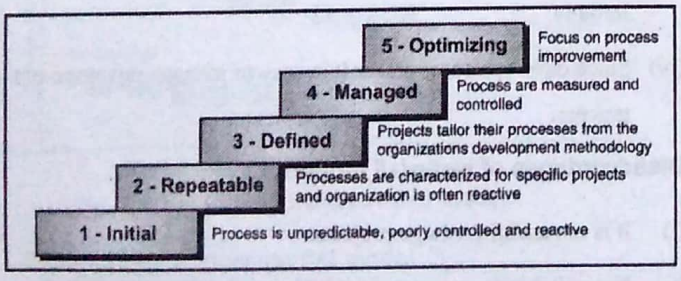


Fig. 1.3 : CMM Levels

1. Level 1 : Initial

This software procedure is stated as inconsistent and occasionally chaotic. Stated procedures and standard practices that present are discarded at the time of crisis. Success of the association mainly depends on efforts taken by each person, capacity, and heroics. The heroes move on to another association taking their knowledge or lessons learnt with them.

2. Level 2 : Repeatable

This level of Software Development association has a fundamental and consistent project management processes to keep the track of cost, schedule, and functionality. The procedure is in region to repeat the latest successes on projects with similar applications. Program management is a most important feature of a level two association.

3. Level 3 : Defined

The software procedure for management and engineering actions are documented, standardized, and incorporated into a standard software procedure for the overall association and each and every project developed by association use an approved, tailored version of the standard software procedure of association for creating, testing and maintaining the software product.

4. Level 4 : Managed

Management can successfully control the efforts required for software development through precise measurements. At this level, association set a quantitative quality aim for both software procedure and software maintenance. At this level, the performance of procedures is controlled through statistical and other quantitative mechanisms, and is quantitatively predictable.

5. Level 5 : Optimizing

The important characteristic of this level is concentrating on repeatedly improving procedure performance with the help of both increasing and innovative technological corrections. At this level, modifications to the procedures are done to increase the procedure performance and at the same time maintaining statistical probability to get the recognized quantitative procedure-improvement aims.

Q. 9 Write note on Capability Maturity Model Integration (CMMI). **May 18**

Ans. : Capability maturity model integration

The CMM model's application in software development has sometimes been problematic. Applying multiple models that are not integrated within and across an organization could be costly in training, appraisals, and improvement activities.

The Capability Maturity Model Integration (CMMI) project was formed to sort out the problem of using multiple models for software development processes, thus the CMMI model has superseded the CMM model, though the CMM model continues to be a general theoretical process capability model used in the public domain.

Scanned by CamScanner

CMMI framework consists of a collection of computer programs based on knowledge, system engineering, software engineering, integrated product and process development and provider sourcing.

CMMI framework has three groups as :

1. CMMI for Development (CMMI - DEV)
2. CMMI for Service (CMMI - SVC)
3. CMMI for Acquisition (CMMI - ACQ)

These three groups forms model components which are uniquely designed for particular business process and they may contain some core processes as a part of them which will be same among these groups. It is also possible to extend CMMI framework by making addition of model component to it.

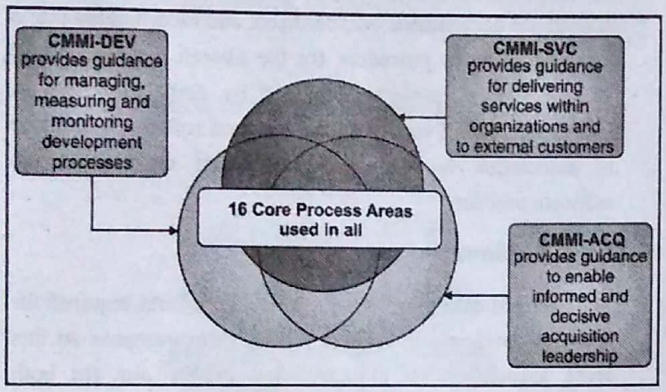


Fig. 1.4 : CMM Framework Groups

Q. 10 Explain waterfall model give its advantage and disadvantages. May 16, Dec. 17

Ans. :

Waterfall model

Waterfall model is the first approach used in software development process. It is also called as classical life cycle model or linear sequential model. In waterfall model any phase of development process begins only if previous phase is completed.

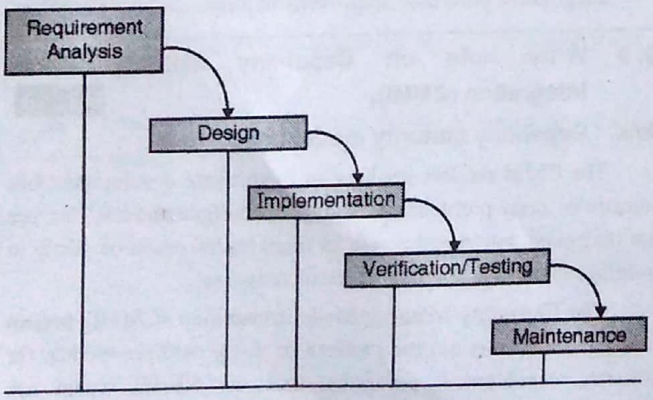


Fig. 1.5 : Waterfall Model

1. **Requirement Analysis**

   In this phase all business requirements of system are gathered and analyzed by communication between stakeholders and customer or user. At the end of this phase Requirement Specification Document (SRS) is created.

2. **Design**

   Based on requirement, specification document design of system is created called software architecture. It is blue print of system representing system's internal structure and behavior.

3. **Implementation**

   In implementation phase actual coding is constructed for software architecture using hardware and software requirements of system. It is responsibility of developer.

4. **Verification / Testing**

   Here coding or job done by developer is verified against requirements of user in order to ensure that software will satisfy all business requirements of user. After the successful verification software is deployed at user's site for their use.

5. **Maintenance**

   While using software if user faces some problems, then those problems must be solved time to time by development team. This task comes under maintenance of software. Maintenance also includes adding new functionalities in software as per user requirements.

Advantages of waterfall model

(i) It is very simple to understand and easy to use.

(ii) Phases of waterfall model do not overlap with each other.

(iii) It is useful for small projects in which requirements are clear initially.

(iv) Since development is linear it is easy to manage development process.

Disadvantages of waterfall model

(i) It is not useful for large projects.

(ii) Not suitable for projects in which requirements are not clear initially.

(iii) System or product is available only at the end of development process.

(iv) It is very difficult to modify system requirements in the middle of development process.

Scanned by CamScanner

Q. 11 Describe incremental model. **Dec. 17**

Ans. : Incremental model

The incremental model applies the waterfall model incrementally. The series of releases is referred to as "increments", with each increment providing more functionality to the customers. After the first increment, a core product is delivered, which can already be used by the customer. Based on customer feedback, a plan is developed for the next increments, and modifications are made accordingly. This process continues with increments being delivered until the complete product is delivered. The incremental philosophy is also used in the agile process model.

Tasks performed by incremental model

1. **Communication :** Helps to understand the objective.
2. **Planning :** Required as many people (software teams) work on the same project but different functions at same time.
3. **Modelling :** Involves business modelling, data modelling, and process modelling.
4. **Construction :** This involves the reuse of software components and automatic code.
5. **Deployment :** Integration of all the increments.

Characteristics of Incremental Model

1. System is broken down into many mini development projects.
2. Partial systems are built to produce the final system.
3. First tackled highest priority requirements.
4. The requirement of a portion is frozen once the incremented portion is developed.

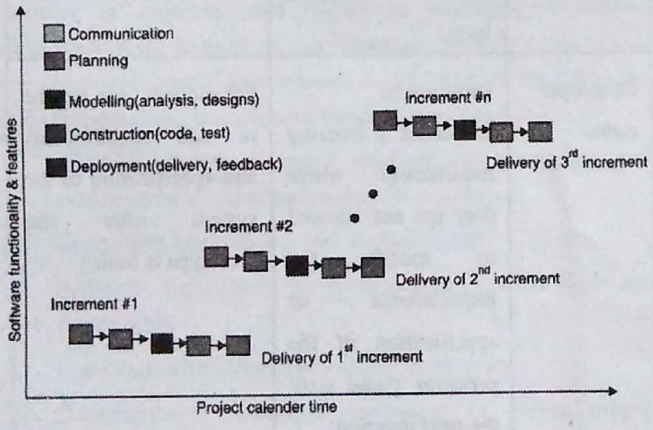


Fig. 1.6 : Incremental Model

Advantages of Incremental Model

1. After each iteration, regression testing should be conducted. During this testing, faulty elements of the software can be quickly identified because few changes are made within any single iteration.
2. It is generally easier to test and debug than other methods of software development because relatively smaller changes are made during each iteration. This allows for more targeted and rigorous testing of each element within the overall product.
3. Customer can respond to features and review the product for any needed or useful changes.
4. Initial product delivery is faster and costs less.

Disadvantages of Incremental Model

1. Resulting cost may exceed the cost of the organization.
2. As additional functionality is added to the product, problems may arise related to system architecture which were not evident in earlier prototypes.

Q. 12 Differentiate between waterfall model and incremental model. **Dec. 17**

Ans. :

Parameter	Waterfall Model	Incremental Model
Simplicity	Simple	Intermediate
Risk Involvement	High	Easily manageable
Flexibility to change	Difficult	Easy
User involvement	Only at beginning	Intermediate
Flexibility	Rigid	Less Flexible
Maintenance	Least	Promotes maintainability
Duration	Long	Very Long

Q. 13 What is evolutionary model? List evolutionary models. **May 18**

Ans. :

Evolutionary model

Normally the Evolutionary Development Model is based on the principle that "stages consist of growing increments of an operational software product, with the way of evolution being discovered by operational experience".

According to the business need and the changing nature of the market there are lot of improvements required in the software product over a time.

Due to this lot of improvement is required in the product hence evolutionary developments model is iterative in nature.

There are two types of models in Evolutionary process. They are :

1. Spiral model
2. Prototyping model

Scanned by CamScanner

Q. 14 Explain in detail Spiral model.

May 15, Dec. 15, May 18

Ans. :

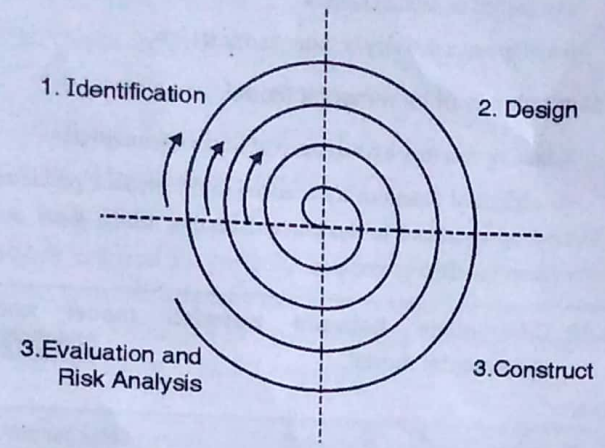


Fig. 1.7 : Spiral Model

Spiral model is a combination of iterative model and waterfall model. Spiral model has four phases of development each of these phases is called as spiral.

(i) Identification

This phase identifies all business requirements of system at the beginning. It involves clear understanding of requirements by communication between stakeholders and customer. In the subsequent iterations all subsystem requirements and unit requirements are identified.

(ii) Design

In first iteration, design phase develops conceptual design of system based on initially gathered requirements. In further spirals or iterations, it develops logical design, physical design, architectural design and final design of system.

(iii) Construct

Initially construct phase develops a code for conceptual design to get user feedback. In next subsequent spirals, detailed working model of software is constructed with increment number and are delivered to customer for feedback.

(iv) Evaluation and risk analysis

In this phase management risks like cost overrun are identified and monitored, technical feasibility of system is also done. At end of each spiral customer evaluates software for potential risks in system and provides feedback. Spiral model can be used for projects where budget and risk evaluation are important factors. If customers are not sure about their requirements then spiral model is useful than waterfall model.

Advantages of spiral model

(i) It is more flexible to changing requirements.

(ii) Requirements are achieved more accurately.

(iii) User can see the system from 1st iteration to end of development.

(iv) Risk management is easier.

Disadvantages of spiral model

(i) It is difficult to manage development process.

(ii) Not useful for small projects development.

(iii) Spiral can run indefinitely.

(iv) It requires excessive documentation work as documentation is prepared for each iteration.

Q. 15 What is difference between Prescriptive and Evolutionary models. **May 15, Dec.16 , May 18**

Ans. :

Parameter	Waterfall Model / Prescriptive Model	Prototype Model / Evolutionary Model
Client preview	Client can preview the system only after the final version of the software is developed because there is no feedback loop.	Client has a preview of the system from the "quick design" and the prototype` is developed early of process.
Developer rights	Developers encounter a freezing requirement where they are not allowed to modify the requirements or specification of the previous phase until the next iteration.	Developers can refine or add requirements and specification to the system after the prototype is built.
Complexity of error	The complexity of an error increases because of the nature of the model; each phase is sequential of the other.	The complexity of an error is low because the prototype enables the developer to detect any deficiency early at the process.
Linearity	Linear	Non-linear

Scanned by CamScanner

Parameter	Waterfall Model / Prescriptive Model	Prototype Model / Evolutionary Model
Suitable for	The waterfall model is better suited for more conventional software projects where user requirements are clear, right from the start.	Prototype model is well suited for online applications where user interfaces are the most important components and clients are not clear about what they exactly need in the final product.

Q. 16 Write note on component based development. **May 16, Dec. 17**

Ans. : Components based development

Commercial off-the-shelf (COST) software components, developed by vendors who offer them as products, can be used when software is to be built. These components provide targeted functionality with well-defined interfaces that enable the component to be integrated into the software. The component-based development model incorporates many of the characteristics of the spiral model. It is evolutionary in nature, demanding an iterative approach to the creation of software.

However, the model composes applications from prepackaged software components. Modeling and construction activities begin with the identification of candidate components. These components can be designed as either conventional software modules or object-oriented classes or packages of classes. Regardless of the technology that is used to create the components, the component-based development model incorporates following steps (implemented using an evolutionary approach) :

1. Available component-based products are researched and evaluated for the application domain in question.
2. Component integration issues are considered.
3. Software architecture is designed to accommodate the components.
4. Components are integrated into the architecture.
5. Comprehensive testing is conducted to ensure proper functionality. The component-based development model leads to software reuse, and reusability provides software engineers with a number of measurable benefits. Based on studies of reusability, component-based development can lead to reduction in development cycle time, reduction in project cost and increase in productivity. Although these results are a function of the robustness of the component library, there is little question that whether the component-based development model provides significant advantages for software engineers.

Q. 17 Explain Formal Methods Model. **Dec. 17**

Ans. : Formal methods model

The formal methods model encompasses a set of activities that leads to formal mathematical specification of computer software. Formal methods enable a software engineer to specify, develop, and verify a computer-based system by applying rigorous, mathematical notation. A variation on this approach is called clean-room software engineering. When formal methods are used during development, they provide a mechanism for eliminating many of the problems that are difficult to overcome using other software engineering paradigms.

Ambiguity, incompleteness, and inconsistency can be discovered and corrected more easily, not through ad hoc review, but through the application of mathematical analysis. When formal methods are used during design, they serve as a basis for program verification and therefore enable the software engineer to discover and correct errors that might otherwise go undetected. Although not a mainstream approach, the formal methods model offers the promise of defect-free software. Yet, concern about its applicability in a business environment has been expressed :

1. The development of formal models is currently quite time-consuming and expensive.
2. Because only few software developers have the necessary background to apply formal methods, extensive training is required.
3. It is difficult to use the model as a communication mechanism for technically unsophisticated customers.

Q. 18 Write note on - Aspect Oriented Model. **May 15, May 17**

Ans. : Aspect oriented development

Regardless of the software process that is chosen, the builders of complex software invariably implement a set of localized features, functions and information content. These localized software characteristics are modeled as components and then constructed within the context of a system architecture. As modern computer-based systems become more sophisticated and complex, certain concerns, customer required properties or areas of technical interest, span the entire architecture. Some concerns are high-level properties of a system; others affect functions or are systemic. When concerns cut across multiple system functions, features, and information they are often referred to as crosscutting concerns.

Aspectual requirements define those crosscutting concerns that have impact across the software architecture. Aspects are mechanisms beyond subroutines and inheritance for localizing the expression of a crosscutting concerns.

Aspect-Oriented Software Development (AOSD), often referred to as Aspect-Oriented Programming (AOP), is a relatively new software engineering paradigm that provides a process and methodological approach for defining, specifying, designing, and

Scanned by CamScanner

constructing aspects. A distinct aspect-oriented process has not yet matured. However, it is likely that such a process will adopt characteristics of both the spiral and concurrent process models.

The evolutionary nature of the spiral is appropriate as aspects are identified and then constructed. The parallel nature of concurrent development is essential because aspects are engineered independently of localized software components and yet aspects have a direct impact on these components.

Q. 19 What is Agility ? Dec.16, Dec.17, May 18

Ans. :

Agile development

Agile software development describes an approach to software development under which requirements and solutions evolve through the collaborative effort of self-organizing and cross-functional teams and their customers (end users). It advocates adaptive planning, evolutionary development, early delivery, and continual improvement, and it encourages rapid and flexible response to change. The term Agile was popularized, in this context, by the Manifesto for Agile Software Development.

The values and principles adopted in this manifesto were derived from and underpin a broad range of software development frameworks, including Scrum and Kanban. There is significant subjective evidence that adopting agile practices and values improves the agility of software professionals, teams and organizations.

Agile software development values

Based on their combined experience of developing software and helping others do that, the seventeen signatories to the manifesto proclaimed that they value:

1. Individuals and Interactions over processes and tools.
2. Working Software over comprehensive documentation.
3. Customer Collaboration over contract negotiation.
4. Responding to Change over following a plan.

As per the view of Scott Ambler (Canadian software engineer, consultant and author) :

1. Tools and processes are important, but it is more important to have competent people working together effectively.
2. Good documentation is useful in helping people to understand how the software is built and how to use it, but the main point of development is to create software, not documentation.
3. A contract is important but is no substitute for working closely with customers to discover what they need.
4. A project plan is important, but it must not be too rigid to accommodate changes in technology or the environment, stakeholders' priorities, and people's understanding of the problem and its solution.

Q. 20 Explain Agile principles. May 17, Dec. 17

Ans. : Agile principles

The Manifesto for Agile Software Development is based on twelve principles :

1. Customer satisfaction by early and continuous delivery of valuable software.
2. Welcome changing requirements, even in late development.
3. Working software is delivered frequently (weeks rather than months).
4. Close, daily cooperation between business people and developers.
5. Projects are built around motivated individuals, who should be trusted.
6. Face-to-face conversation is the best form of communication (co-location).
7. Working software is the primary measure of progress.
8. Sustainable development, able to maintain a constant pace.
9. Continuous attention to technical excellence and good design.
10. Simplicity is essential.
11. Best architectures, requirements, and designs emerge from self-organizing teams.
12. Regularly, the team reflects on how to become more effective, and adjusts accordingly.

Q. 21 What are Advantages of Agile Processes? Dec. 17

Ans. : Advantages of agile processes

There are number of advantages of Agile Process

1. Stakeholder Engagement,
2. Transparency,
3. Early and Predictable Delivery,
4. Predictable Costs and Schedule,
5. Allows for Change,
6. Focuses on Business Value,
7. Focuses on Users,
8. Improves Quality.

Q. 22 Is agile process suitable for large scale projects ? Mention few issues. Dec. 15

Ans. :

Agile methodology is not considered as suitable for large projects because of following important issues :

1. Lack of clarity
2. Continual reliance on legacy methods
3. Inadequate experience with agile
4. Lack of collaboration in teams composed by different companies
5. Lack of a Testing Strategy
6. Lack of alignment in other areas of the enterprise
7. Larger teams and big pyramid structures
8. Not changing the objectives.

Scanned by CamScanner

Q. 23 Explain SCRUM. May 15

Ans. : SCRUM

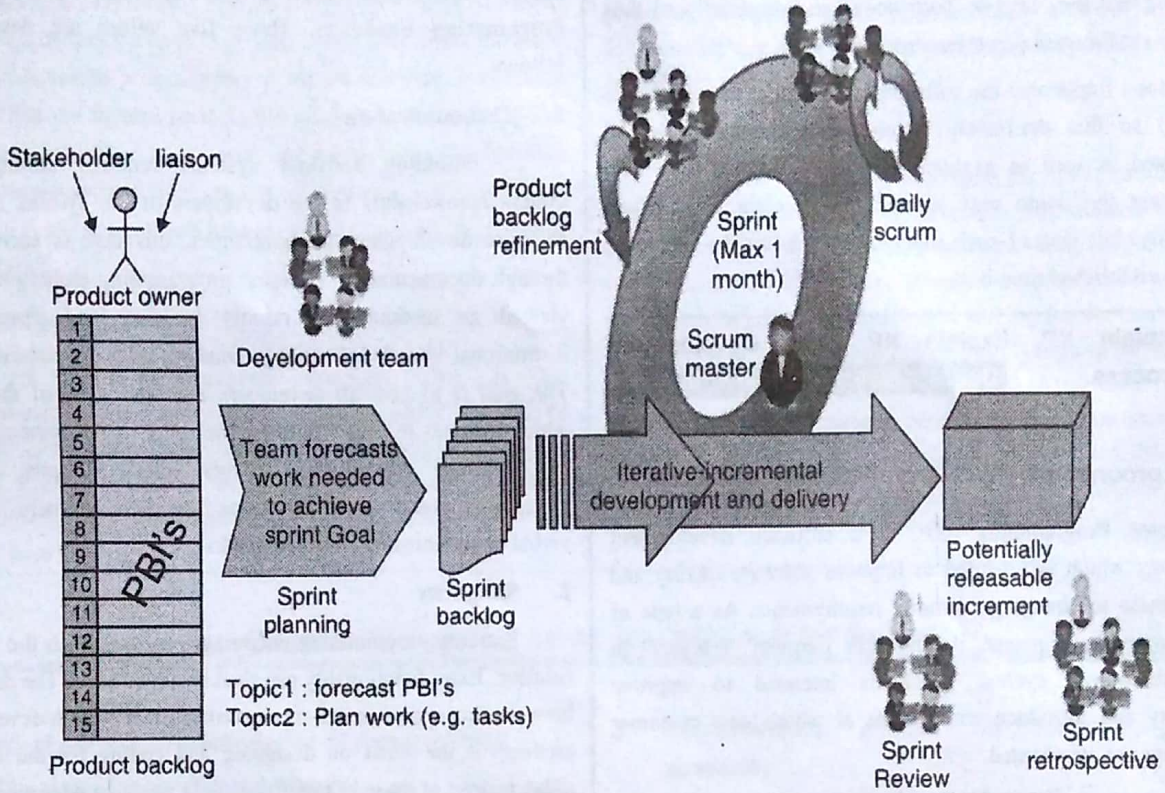


Fig. 1.8 : Scrum Framework

Scrum is an agile framework for managing knowledge work, with an emphasis on software development. It is designed for teams of three to nine members, who break their work into actions that can be completed within time boxed iterations, called "sprints", no longer than one month and most commonly two weeks, then track progress and re-plan in 15-minute stand-up meetings, called daily scrums. Scrum principles are mostly based on the agile manifesto and help to guide activities regarding development inside a process which includes framework activities such as requirements, analysis, design, evolution, and delivery. In all the framework activities, work tasks happen within a process pattern called a sprint. The work implemented in a sprint is tailored to the problem at hand and is defined and usually tailored in real time by the respective Scrum team.

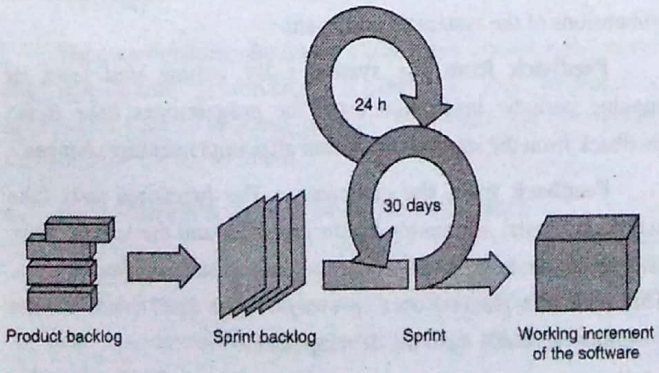


Fig. 1.9 : The scrum process

Scrum mostly focus on the use of a bunch of software process patterns which are proved effective for projects with restrictive timelines, changing requirements, and business criticality. All such process patterns defines several sets of development actions :

Backlog : A prioritized list regarding project needs or features which offer business value for the customer. It is possible to add items to the backlog at any time. The respective product manager assesses the backlog and changes the priorities as per necessity.

Sprints : Includes work units which are necessary to gain a requirement defined in the backlog which should be incorporated in predefined time-box (usually thirty days). Modifications (such as backlog work items) are not involved during the sprint. Therefore, team members are allowed by the sprint to work in a short-term, but stable environment. Scrum meetings - are short (usually fifteen minutes) meetings conducted on daily basis by the Scrum team. Three most important key questions which are asked and must be answered by all team members are :

1. What did you do since the last team meeting ?
2. What obstacles are you encountering ?
3. What do you plan to accomplish by the next team meeting ?

Scanned by CamScanner

A team leader, who is known as a Scrum master, handles the meeting and evaluates the responses from each person. The important use of Scrum meeting to the team is that it can uncover critical issues as early as possible. One more benefit of these daily meetings is that they lead to "knowledge socialization" and thus encourage a self-organizing team structure.

Demos : Implement the software increment at the client side (customer) so that the newly developed functionality can be demonstrated as well as evaluated by the customer. It is also possible that the demo may not able to contain all planned functionality, but instead such functions which can be delivered within the established time-box.

Q. 24 Explain XP. Explain XP Agile Development Process. **Dec. 16, May 17, Dec. 17, May 18**

Ans. :

Extreme programming

Extreme Programming (XP) is a software development methodology which is intended to improve software quality and responsiveness to changing customer requirements. As a type of agile software development, it advocates frequent "releases" in short development cycles, which is intended to improve productivity and introduce checkpoints at which new customer requirements can be adopted.

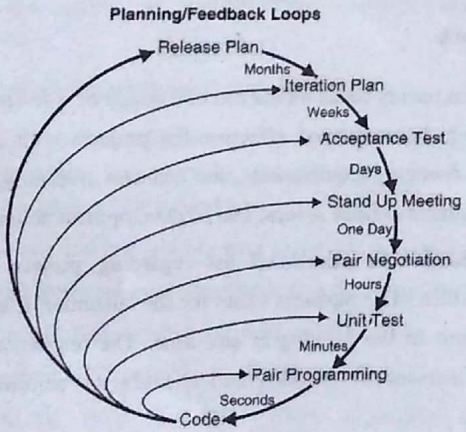


Fig. 1.10 : Extreme Programming

Other elements of extreme programming include: programming in pairs or doing extensive code review, unit testing of all code, avoiding programming of features until they are actually needed, a flat management structure, code simplicity and clarity, expecting changes in the customer's requirements as time passes and the problem is better understood, and frequent communication with the customer and among programmers. The methodology takes its name from the idea that the beneficial elements of traditional software engineering practices are taken to "extreme" levels. As an example, code reviews are considered a beneficial practice; taken to the extreme, code can be reviewed continuously, i.e. the practice of pair programming.

XP Values

Extreme programming initially recognized four values in 1999 : communication, simplicity, feedback, and courage. A new value, respect, was added in the second edition of Extreme Programming Explained. Those five values are described as follows

1. Communication

Building software systems requires communicating system requirements to the developers of the system. In formal software development methodologies, this task is accomplished through documentation. Extreme programming techniques can be viewed as methods for rapidly building and disseminating institutional knowledge among members of a development team. The goal is to give all developers a shared view of the system which matches the view held by the users of the system. To this end, extreme programming favors simple designs, common metaphors, collaboration of users and programmers, frequent verbal communication, and feedback.

2. Simplicity

Extreme programming encourages starting with the simplest solution. Extra functionality can then be added later. The difference between this approach and more conventional system development methods is the focus on designing and coding for the needs of today instead of those of tomorrow, next week, or next month. This is sometimes summed up as the "You aren't gonna need it" (YAGNI) approach.

Proponents of XP acknowledge the disadvantage that this can sometimes entail more effort tomorrow to change the system; their claim is that this is more than compensated for by the advantage of not investing in possible future requirements that might change before they become relevant. Coding and designing for uncertain future requirements implies the risk of spending resources on something that might not be needed, while perhaps delaying crucial features. Related to the "communication" value, simplicity in design and coding should improve the quality of communication. A simple design with very simple code could be easily understood by most programmers in the team.

3. Feedback

Within extreme programming, feedback relates to different dimensions of the system development :

Feedback from the system : By writing unit tests, or running periodic integration tests, the programmers have direct feedback from the state of the system after implementing changes.

Feedback from the customer : The functional tests (aka acceptance tests) are written by the customer and the testers. They will get concrete feedback about the current state of their system. This review is planned once in every two or three weeks so the customer can easily steer the development.

Scanned by CamScanner

Feedback from the team : When customers come up with new requirements in the planning game the team directly gives an estimation of the time that it will take to implement.

Feedback is closely related to communication and simplicity. Flaws in the system are easily communicated by writing a unit test that proves a certain piece of code will break. The direct feedback from the system tells programmers to recode this part. A customer is able to test the system periodically according to the functional requirements, known as user stories. As per Kent Beck (American software engineer and the creator of extreme programming), "Optimism is an occupational hazard of programming. Feedback is the treatment".

4. Courage

Several practices represent courage. One is the commandment to always design and code for today and not for tomorrow. This is an effort to avoid getting delayed in design and requiring a lot of effort to implement anything else. Courage enables developers to feel comfortable with refactoring their code when necessary.

This means reviewing the existing system and modifying it so that future changes can be implemented more easily. Another example of courage is knowing when to throw code away : courage to remove source code that is obsolete, no matter how much effort was used to create that source code. Also, courage means persistence : a programmer might be stuck on a complex problem for an entire day, then solve the problem quickly the next day, but only if they are persistent.

5. Respect

The respect value includes respect for others as well as self-respect. Programmers should never commit changes that break compilation, that make existing unit-tests fail, or that otherwise delay the work of their peers. Members respect their own work by always striving for high quality and seeking for the best design for the solution at hand through refactoring. Adopting the four earlier values leads to respect gained from others in the team. Nobody on the team should feel unappreciated or ignored. This ensures a high level of motivation and encourages loyalty toward the team and toward the goal of the project. This value is dependent upon the other values, and is oriented toward teamwork.

Chapter 2 : Requirements Analysis and Modelling

Q. 1 What are the characteristics of requirements ?

Ans. : Characteristics of requirements

There are various characteristic of requirements. They are as follows :

1. Requirements should be unambiguous

Ambiguous means the single word or statement has more than one meaning. If requirements contain ambiguity then it is difficult to fulfill the requirements correctly. Therefore the requirements should be unambiguous means non confusing.

2. Requirements should be testable (verifiable)

The requirements should be testable means tester should be able to easily test the requirement whether they are implemented successfully or not. For easy test, the requirement should be clear and unambiguous.

3. Requirements should be clear (concise, simple, precise)

The requirements should be clear. They should not contain any unnecessary information. If there is any unnecessary information then it becomes difficult to achieve the appropriate fulfillment of the requirement.

4. Requirements should be understandable

The requirements should be understandable means when anyone read it then it should be easily understood by that person. To make it understandable proper conventions are used. It should be grammatically correct.

5. Requirements should be feasible (realistic, possible)

The requirement should be completed within the given time and budget. Specified requirement said to be feasible if it can be implemented using existing technology, with estimated budget and time.

6. Requirements should be Consistent

Consistency is an important feature of requirements. It means that all inputs must be processed similarly. It should not happen that processes produce different outputs for same inputs coming from different sources.

Q. 2 Explain the types of requirements.

Ans. : Types of requirements

The requirements, which are in general assumed, are divided into three categories, namely, functional requirements, non-functional requirements, and domain requirements.

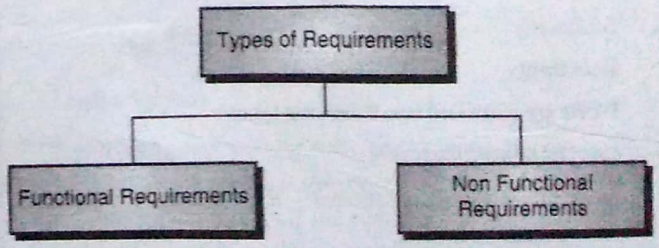


Fig. 2.1 : Types of requirements

Scanned by CamScanner

Functional Requirements

The Functional requirements specification create document regarding the operations as well as activities which a system must be able to carry out. In software engineering, a functional requirement is used to define a function regarding a system or its component, in which a function is described as a specification of behavior between outputs and inputs. In the functional requirements there may be elements such as calculations, technical details, data manipulation and processing, and other specific functionality which define what a system is supposed to implement.

Functional requirements is used to mention specific results of a system. This should be contrasted with non-functional requirements that mention overall characteristics like as cost and reliability. Functional requirements drive the application architecture of a system, while non-functional requirements drive the technical architecture of a system.

Functional requirements should include descriptions of data to be entered into the system. Descriptions of operations performed by each screen. Descriptions of work-flows performed by the system. Descriptions of system reports or other outputs. Persons who can enter the data into the system. Way the system meets applicable regulatory requirements.

Example of functional requirements

Functional requirements must contain functions performed by specific screens, outlines of work-flows performed by the system, and other business or compliance requirements the system must meet.

Interface requirements

1. Field 1 accepts numeric data entry.
2. Field 2 only accepts dates before the current date.
3. Screen 1 can print on-screen data to the printer.

Non Functional Requirements (NFR)

These are basically the quality constraints that the system must satisfy according to the project contract. The priority or extent to which these factors are implemented varies from one project to other. They are also called non-behavioral requirements. They basically deal with issues like :

1. Portability
2. Security
3. Maintainability
4. Reliability
5. Scalability
6. Performance
7. Reusability
8. Flexibility

NFRs are classified into following types :

(a) Interface constraints

(b) Performance constraints: response time, security, storage space, etc.

(c) Operating constraints

(d) Life cycle constraints : maintainability, portability, etc.

(e) Economic constraints

The process of specifying non-functional requirements needs the knowledge of the functionality of the system, and also the knowledge of the context within which the system will operate.

Types of Non-Functional Requirements

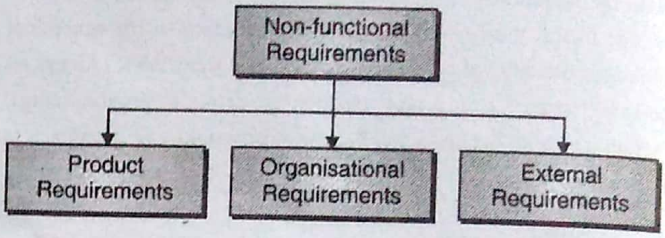


Fig. 2.2 : Types of non-functional requirements

Product requirements : Requirements which mention that the delivered product should have behavior in a specific way, e.g. execution speed, reliability etc.

Organizational requirements : Requirements that are a result of organizational policies as well as procedures, such as process standards which has been used, implementation necessities etc.

External requirements : Requirements which are generated from the factors that are external to the system and its development process, e.g. interoperability requirements, legislative requirement.

Q. 3 What are different requirements engineering tasks? May 16, Dec. 16, May 17 May 18,

Ans. : Requirement engineering tasks

The activities/tasks involved in requirements engineering vary widely, depending on the type of system being developed and the specific practices of the organization(s) involved.

These may include :

1. Requirements inception

Inception is a task where the requirement engineering asks a set of questions to establish a software process. In this task, it understands the problem and evaluates with the proper solution. It collaborates with the relationship between the customer and the developer. The developer and customer decide the overall scope and the nature of the question.

2. Requirements elicitation

In requirements engineering, requirements elicitation is the practice of researching and discovering the requirements of a system from users, customers, and other stakeholders. The practice is also sometimes referred to as "requirement gathering". The term elicitation is used in books and research to raise the fact that good requirements cannot just be collected from the customer, as would be indicated by the name requirements gathering. Requirements elicitation is non-trivial because you can never be sure you get all

Scanned by CamScanner

requirements from the user and customer by just asking them what the system should do OR NOT do (for Safety and Reliability).

Requirements elicitation practices include interviews, questionnaires, user observation, workshops, brainstorming, use cases, role playing and prototyping. Before requirements can be analyzed, modeled, or specified they must be gathered through an elicitation process. Requirements elicitation is a part of the requirements engineering process, usually followed by analysis and specification of the requirements.

3. Requirements analysis and negotiation

Requirements are identified (including new ones if the development is iterative) and conflicts with stakeholders are solved. Both written and graphical tools (the latter commonly used in the design phase but some find them helpful at this stage, too) are successfully used as aids. Examples of written analysis tools : use cases and user stories.

Examples of graphical tools : UML and LML.

4. System modelling

Some engineering fields (or specific situations) require the product to be completely designed and modeled before its construction or fabrication starts and, therefore, the design phase must be performed in advance. For instance, blueprints for a building must be elaborated before any contract can be approved and signed. Many fields might derive models of the system with the Lifecycle Modeling Language, whereas others, might use UML.

5. Requirements specification

Requirements are documented in a formal artifact called Requirements Specification (RS). Nevertheless, it will become official only after validation. A RS can contain both written and graphical (models) information if necessary.

Example : Software Requirements Specification (SRS).

6. Requirements validation

It is the process of checking whether the documented requirements and models are consistent and meet the needs of the stakeholder. Only if the final draft passes the validation process, the RS becomes official.

7. Requirements management

Managing all the activities related to the requirements since inception, supervising as the system is developed and, even until after it is put into use (e. g., changes, extensions, etc.)

Q. 4 Mention any four requirement Elicitation Methods.

Ans. : Requirement elicitation

In requirements engineering, requirements elicitation is the practice of researching and discovering the requirements of a system from users, customers, and other stakeholders. The practice is also sometimes referred to as "requirement gathering".

Different ways to identify customer requirement

1. Interviews

Interviews are important way for gathering requirements. Organization may take various kinds of interviews such as :

Structured or closed interviews in which information to be collected from customer is decided in advance. Non-structured or open interviews in which details needs to be collected from customer are decided in advance.

(a) Oral interviews.

(b) Written interviews.

(c) Face-to-face interviews which are held among two people across the table.

(d) Groups of persons are participating in group interviews. They support to cover any missing requirement as number of people participates in this process.

2. Surveys

Organization may perform surveys of different stakeholders by questioning them about their requirements and expectation from the proposed software system. The advantage of this technique is that, collecting requirements is economically beneficial because it collects requirements from a large number of persons at same time. Surveys are less effective method of data discovery. Frequently, a survey's main finding is that other questions should have been asked instead. Surveys are most useful for capturing clear factual information.

3. Questionnaires

Questionnaires are a document way which contains in-built set of objectives that is based on questions and their options. This document is given to all stakeholders to give answer of those questions which are gathered and compiled. Output of this mechanism is, if an answer for some question is not given in the questionnaire, the issue may be remaining unattended. The advantage of this technique is that, collecting requirements economically is beneficial because it collects requirements from huge amount of persons at same time. Questionnaires are less effective method of data discovery.

4. Task analysis

In this technique, team of software developers and engineers may identify the functional specifications for which the new system is needed to develop. If the customer has some software to do the particular operations then it is analyzed by this team to find out requirements for proposed system.

Scanned by CamScanner

Q. 5 How important is Requirements analysis?

Dec. 16, May 18

Ans. : Importance of requirement analysis

Requirements analysis is vital phase of requirement engineering process after Elicitation. In this activity, we understand and refine the collected requirements to make them consistent and non-confusing. This activity reviews each and every requirement and may also give graphical view of the overall system. After the analysis, it is supposed that the understanding of requirments of project are increased significantly. In this activity, developer can communicate with customer to clear confusing points and to understand which requirements are more vital than others. The significant factors in the requirement analysis activity are :

1. Identify and solve confusion among requirements at the same level and among different levels.
2. Identify the boundaries of the proposed software system and the way in which it communicate with environment.
3. Evaluate customer requirements for overall system requirements and then break down them into individual component level.

The objective of the Requirement Analysis is generating a solution to implement the requirements. Process of the Requirements Analysis includes following steps :

1. Analysis of the requirements
2. Description of the solution
3. Cost estimate and prioritization.

Here are some examples of how we can consider problems occurred in requirement analysis. Stakeholders don't recognize what they really required. Stakeholders state the requirements in their own words. Various stakeholders may have different and confusing requirements. Organizational and political factors may affect the software needs. The requirements are modified at the time of requirement analysis process. New stockholders may be included and the business environment may change.

Q. 6 What is SRS document ? State advantages of it.

Ans. :

Software requirement specification

SRS stands for Software/System Requirement Specification. SRS is a special kind of document which contains user requirements for a system which states properties and constraints that must be satisfied by a software system.

IEEE defines software requirements specification as, 'a document that clearly and precisely describes each of the essential requirements (functions, performance, design constraints and quality attributes) of the software and the external interfaces.

The outcome of the requirements gathering and analysis phase of the SDLC is Software Requirements Specification (SRS).

SRS is also called as requirements document. SRS is base for software engineering actions and is generated when all requirements of software project are gathered and analyzed. It is generally signed at the end of requirements engineering phase. Following are 6 requirements stated by Heninger, which SRS document should follow :

1. SRS document should specify external system behaviour.
2. SRS document should specify implementation constraints.
3. It should be easily changeable if any changes occur.
4. It should act as reference tool for maintaining the system.
5. SRS document record forethought about the lifecycle of the system i.e. predicts changes.
6. It must include acceptable response to undesired events.

Advantages of SRS

1. SRS contains the base for agreement among the client and the company who develop the product as per expectations.
2. A SRS gives a base for verification of the completely developed product.
3. Using high-quality SRS, we can develop high-quality software product.
4. A high-quality SRS decreases the cost of development.

Q. 7 Explain any five characteristics of SRS.

Ans. :

A good software requirement specification must satisfy following characteristics.

1. Correct

SRS is correct when all requirements of user are described in the requirement document. The listed requirements must be matched with desired system. This depict that every requirement is analyzed to give assurance that it (SRS) contains user requirements. Remember that there is no specific tool or process to ensure the correctness of SRS. Correctness gives assurance that all stated requirements are worked as expected.

2. Unambiguous

SRS does not contain any confusion when each specified requirement has single interpretation. This characteristic state that every requirement is individually interpreted. In situation, where one term has number of meanings, then its meaning must be specified in the SRS so that it will be non-confusing and simple to understand.

3. Complete

SRS is complete when the requirements undoubtedly specified that which work the software product is required to perform. This contains each and every requirement associated to performance, design and functionality.

Scanned by CamScanner

4. Ranked for importance/stability

Each and every requirement has not same importance, so every requirement is recognized to make differentiation between requirements. For this purpose, it is needed to undoubtedly recognize every requirement. Stability refers to the probability of further modifications in the requirement.

5. Modifiable

The requirements given by user are changeable, so requirement document must be generated in such a way that those modifications can be included easily in SRS by preserving consistently the structure and style of the SRS.

6. Traceable

SRS is observable when the source of every requirement is unambiguous and facilitates the description of every requirement in future. Forward tracing and backward tracing methods are used for this purpose. Forward tracing state that every requirement must be referencing to design and code of components. Backward tracing state every requirement explicitly addressing its source.

7. Verifiable

SRS is testable when the stated requirements can be tested with a cost-effective procedure to verify whether the final software fulfills those requirements. The requirements are tested through reviews. Remember that clear requirement is needed for verifiability.

Q. 8 Explain different metrics for size estimation. **May 18**

Ans. : Metrics for size estimation

Size oriented software metrics are in general derived by the process of normalizing quality and/or productivity measures by assuming the software size which has been produced. If a software organization is able to keep simple records, a table regarding size-oriented measures will be as shown in Table 2.1.

In Table 2.1, the details shown about all the software projects are developed in last five years.

Project alpha : 12,100 lines is the total size of code with 24 person-months of effort at a price of $168,000.

Table 2.1

Project	LOC	Effort	$(000)	Pp doc	Errors	Defects	People
alpha	12,100	24	168	365	134	29	3
beta	27,200	62	440	1224	321	86	5
gamma	20,200	43	314	1050	256	64	6

It is important that the effort and cost recorded in the table does not represent only coding, but it also represents various software engineering activities such as analysis, design and testing. The further information shows that the size of documentation is of 365 pages. Before the release of software the count of errors was 134. Within the first year after release to the customer, the count of defects was twenty nine. The number of people involved in development was three. For the purpose of metrics development which can be assimilated with other same types of metrics from various projects, as our normalization value, the Line of code can be selected. As per the table information, a set of simple size-oriented metrics can be generated for all the projects :

1. Errors per KLOC (Thousands line of code)
2. Defects per KLOC,
3. $ Per KLOC,
4. Pages of documentation per KLOC.

The Size-oriented metrics are not considered as the best solution for measuring the software process.

Line of Code (LOC)

LOC (Line of Code) is considered as the most simplest among all metrics which are available for the purpose of estimating project size. Because of its simplicity, this metric is very popular. With the help of this metric, it is possible to estimate the project size by the way of counting the number of source instructions present in the respective developed program. It is most likely that in the process of counting the number of source instructions, lines which comment the code as well as the header lines will be ignored. At the end of a project it is very easy to determine the LOC count. However, in initial stage, exact estimation regarding the LOC count of a project is not easy.

For the purpose of estimating the LOC count in the initial phase of a project, project managers generally split the problem into modules, and further those modules into submodules and so on, unless and until it is not possible to approximately predict the sizes of the different leaf-level modules. Experience of developing similar products is useful to be able to do this. By taking the reference of the estimation of the lowest level modules, total size estimation is done by the project managers.

FP (Function Points) Based Estimation

As a normalization value, a measure of the functionality which is given by the application is taken by the Function-oriented software metrics. It is not possible to measure the functionality directly. Hence using other direct measures, it is necessary to derive it indirectly. The concept of Function-oriented metrics was proposed initially by Albrecht. A measure called the function point is suggested by him. Function points are basically derived with the help of an empirical relationship depending upon countable (direct) measures regarding the information domain of software as well as software complexity assessments. Five information domain characteristics are determined and counts are provided. These are as follows :

Scanned by CamScanner

1. **Number of user inputs :** Each user input should be considered
2. **Number of user outputs :** Reports, screens, error messages, etc
3. **Number of user inquiries :** An on-line i/p which results on on-line o/p
4. **Number of files :** Each logical master file is counted
5. **Number of external interfaces :** Data files on storage media

Once the collection of these data is completed, each count is assigned a complexity value. Enterprises which refer function point methods establish criteria for the purpose of determining whether a specific entry is simple, average, or complex.

Computing function points	Measurement parameter	Count		Weighting factor Simle	Average	Complex		
	Number of inputs		x	3	4	6	=	
	Number of user outputs		x	4	5	7	=	
	Number of user inquiries		x	3	4	6	=	
	Number of files		x	7	10	15	=	
	Number of external interfaces		x	5	7	10	=	
	Count total							

Fig. 2.3

Use case based estimation

The UCP (Use Case Points) is a software estimation technique which is used to make forecast regarding the software size for the respective software development projects. UCP is helpful in the situations when the UML (Unified Modeling Language) and RUP (Rational Unified Process) techniques are being used for the the purpose of software design as well as development. The terminology of Use Case Points is depend upon the requirements for the particular system being written with the help of use cases, which is an integral element of the UML set of modeling techniques. The calculation of UCP (software size) is done as per the elements of the system related use cases with factoring to account for the considerations related to technical and environmental aspects. Then it is possible to use the UCP for a project for the purpose of calculating the estimated effort of the particular project. The method to calculate the size estimate for development of a system depends upon a calculation with the given elements :

1. **UUCW (Unadjusted Use Case Weight) :** The point size of the software which is concerned with the number as well as complexity of use cases.
2. **UAW (Unadjusted Actor Weight) :** The point size of the software which is concerned with the number as well as complexity of actors.
3. **TCF (Technical Complexity Factor) :** Factor which helps to adjust the size depending upon technical considerations.
4. **ECF (Environmental Complexity Factor) :** Factor which helps to adjust the size depending upon environmental considerations.

After calculation of all the four elements, it is possible to calculate the final size estimate. This end result is called as the Use Case Points or UCP for a software development project.

Chapter 3 : Analysis and Design Engineering

Q. 1 What is Design ? Dec. 16

Ans. : Design

Once the requirements document regarding the software to be developed is presented, the phase of software design gets started. The requirement specification activity is considered purely related with the problem domain whereas design is considered as the initial phase of transforming the problem into a solution. In the design phase, all the relevant entities such as the customer, business requirements and technical considerations collaborate to formulate a product or a system.

In design process there are several elements such as set of principles, concepts and practices, which help a software engineer to model the system or product which is to be built. The design model is assessed for quality and reviewed before generation of code and execution of tests. The design model gives detailed information regarding software data structures, architecture, interfaces and components which are necessary to employ the system.

Q. 2 Explain qualities of good software design.

Ans. : Qualities of good software design

There are number of qualities of good design :

(1) Innovative

Innovative design can be either completely new design or redesign of existing product. New design gives unseen value to market where as redesign improves the quality of an existing product.

(2) Functional

Good design fulfils all its intended functions to solve user's problem. It focuses on usefulness of a product by optimizing its functionality.

(3) Honest

A good design is honest. An honest design expresses the functions and values it offers. It never attempts to modify Organiser's and User's view with promises it can't keep.

Scanned by CamScanner

(4) User - oriented

Good design is developed based on its use and intended to improve solution to problem for its user. User-oriented design gives intellectual as well as material value to system which in turn achieves user's satisfaction.

(5) Correctness

Correctness is an important quality of good design. A good design should correctly achieve all required functionalities as per SRS document.

Q. 3 Explain design principles. **Dec. 16**

Ans. : Design principles

Following are important design principles :

1. The design process should not suffer from 'tunnel vision'.
2. The design should be traceable to the analysis model.
3. The design should not reinvent the wheel.
4. The design should "minimize the intellectual distance" between the software and the problem as it exists in the real world.
5. The design should exhibit uniformity and integration.
6. The design should be structured to accommodate change.
7. The design should be structured to degrade gently, even when unusual data, events, or operating conditions are encountered.
8. Design is not coding, coding is not design.
9. The design should be assessed for quality as it is being created, not after the creation.
10. The design should be reviewed to minimize conceptual (semantic) errors.

Q. 4 Explain software design concepts. **May 16, May 17, Dec 17, May 18**

Ans. : Software design concepts

All of the software processes are characterized by basic concepts in conjunction with some practices or methods. Methods represent the way by which the concepts are applied. An Old technology is replaced by new technology, several modifications occur in the methods which are used to apply the concepts for the process of developing the software. However, the fundamental concepts which underline the software design process always remain the same, These concepts are :

(A) Abstraction

Abstraction refers to a process by which software designers consider components at an abstract level, at the same time as ignoring the implementation details of the components.

IEEE defines abstraction as 'a view of a problem that extracts the essential information relevant to a particular purpose and ignores the remainder of the information.'

There are two ways to use the concept of abstraction; as a process and as an entity.

As a **process,** abstraction refers to a mechanism of hiding irrelevant extra details and representing just the important essential features of an element so that one can concentrate on essential things at a time. As an entity, abstraction refers to a model or view of an item. In the software process all the steps are accomplished via several levels of abstraction. At the highest level, one can get the outline of the solution to the respective problem while at the lower levels, he/she will get the solution to the problem in detail.

For example, in the requirements analysis phase, language is used to provide a solution to the problem and as one proceeds during the software process, the abstraction level reduces and source code of the software is generated at the lowest level. In general three abstraction mechanisms used in software design are functional abstraction, data abstraction and control abstraction. These mechanisms help to control the complexity of the design process by starting from the abstract design model and ending at concrete design model in a systematic manner.

1. **Functional abstraction :** This type of abstraction includes the use of parameterized subprograms. Functional abstraction can be created as a set of subprograms known as 'groups'. In these sets, routines are exist which may be visible or invisible. The use of visible routines can be done within the containing groups and also within other groups, while invisible routines are hidden from other groups and are allowed to be used in the containing group only.
2. **Data abstraction :** This type of abstraction includes describing data which specifies a data object. For example, the data object *flower* encompasses a set of attributes (color, fragrance) which describe the flower object clearly. In this abstraction mechanism, representation as well as manipulation details are ignored.
3. **Control abstraction :** This type of abstraction states the desired effect, without conducting the exact mechanism of control. For example, in programming language like C, *if* and *while* statements are abstractions of machine code implementations containing conditional instructions.

(B) Information Hiding

It is important to specify and design modules in such a manner that the data structures as well as processing details of one module should not be accessible to any other modules. Only required information is transferred between the modules. The way of hiding unnecessary details is referred to as information hiding.

IEEE defines information hiding as 'the technique of encapsulating software design decisions in modules in such a way that the module's interfaces reveal as little as possible about the module's inner workings; thus each module is a 'black box' to other modules in the system.

Scanned by CamScanner

Information hiding is an important aspect when there is need of modifications during the testing and maintenance phase.

Advantages of information hiding

1. Results in low coupling.
2. Put emphasis on communication by controlled interfaces.
3. Reduces the possibility of adverse effects.
4. Controls the impacts of changes in one component to others.
5. Results in higher quality software.

(C) Structure / Architecture

Software architecture refers to the structure of the system, which consists of several components regarding a program/ system, the attributes (properties) of those components and the relationship among them. The software architecture helps the software engineers in the process of analyzing the software design proficiently. Additionally it also helps the software engineers in decision-making as well as handling risks. Following tasks are implemented by the software architecture :

1. Give an insight to all the relevant stakeholders which helps them to communicate with each other.
2. Spot the early design decisions having big impact on the software engineering activities such as coding and testing which are followed after the design phase.
3. Generates intellectual models regarding the way by which the system is organized into components and the way by which these components communicate with each other.

(D) Modularity

Modularity can be achieved by the process of dividing the software into components which are uniquely named and addressable. These components are also known as modules. A complex system (huge application) is partitioned into a set of discrete modules in such a manner that one should be able to develop those modules independent of each other. After development is completed, the modules are integrated to form an entire project. One has to remember that as the number of modules a system is divided into increases, it becomes easy to handle the software but it also increases the effort required to integrate the modules.

The process of modularizing a design helps to plan the development in a more efficient way, accommodate changes without difficulty, conduct testing and debugging effectively as well as efficiently, and carry out the maintenance work without adversely affecting the functionality of the software.

Example

1. The ubiquitous television set is an example of a system made up of a number of modules – speakers, projection tube, channel buttons, volume buttons, etc.
2. Each module has its own defined functionality but when they are put together synergistically, the complete functionalities of a television are realized.

(E) Concurrency

It is important to utilize the resources efficiently as much as possible. For this purpose multiple tasks must be executed concurrently. This aspect makes concurrency one of the important concepts of software design. Every system should be designed in such a manner that it should facilitate multiple processes to execute concurrently. For example, if the currently executing process is waiting for some resource, the system must be able to execute any other process in the mean time.

(F) Verification

Verification is described as a mechanism of confirmation by examining and giving evidence that there is no mismatch in design output and the design input specifications. Design input can be any physical as well as performance requirement which is used as the basis for the purpose of designing. Design output is considered as the result of all the design phases' efforts. The final design output is a basis for device master record.

(G) Aesthetics

Aesthetics is the philosophical study of beauty and taste.

Software engineering is a largely creative design discipline, where new designs (both graphical design models as well as code) are created from scratch. Compared to art, there are many targets (e.g. desired functionality) and constraints (e.g. desired quality attributes) that the software engineer must adhere to, but these still leave almost infinitely many ways to create a model that fulfills all those requirements.

In addition, for essentially the same model, there are many ways to represent it. Since models and code must also serve to communicate ideas to other people (or to the same people months or years later), the chosen structure and representation are key attributes for the success of the product. It's a well-accepted hypothesis that the aesthetics of the artifacts produced during software development are an indication of 'the quality': it is very common to comment on designs as being 'beautiful" or "elegant". Such a qualification is used to indicate the apparent quality of the design. Apparently, we make an intuitive connection between aesthetics and quality.

Q. 5 Explain cohesion. **Dec. 16, May 17**

Ans. :

Cohesion

Cohesion define the degree to which components of system are related to each other i.e. it measures the strength of relationship between two components of system. Good system design must have high cohesion between the components of system.

Scanned by CamScanner

(i) Co-incidental cohesion

An unplanned cohesion which results into decomposition of a program into smaller components for the modularization of system is called co-incidental cohesion. Typically these types of cohesions are never accepted.

(ii) Logical cohesion

When logically classified elements are combined into a single module then it is called as logical cohesion. Here all elements of modules contribute to same system operation.

(iii) Temporal cohesion

Temporal cohesion occurs when component of a system performs more than one function and these functions must occur within same time span. It is generally found in initiation and termination activities.

(iv) Procedural cohesion

When components of system are related to each other only by sequence then there exists procedural cohesion. Loop in programming language is good example of procedural cohesion.

(v) Communication cohesion

When elements of module perform different functions but each function accepts same input and generates same output then that module is said to be communicational cohesive. It is sometimes acceptable if no other alternative is easily found.

(vi) Sequential cohesion

A module is said to be sequentially cohesive if its functions are related in a way such that output of one function acts as an input data to other function. This type of cohesion is easily maintainable.

(vii) Functional cohesion

If elements of module are grouped together since they contribute to a single function then such a module is functionally cohesive module. All elements of such a module are necessary for successful execution of function.

Q. 6 Explain coupling. Dec. 16, May 17

Ans. : Coupling

Coupling is an indication of strength of interconnection between various components of a system. Highly coupled components are more dependent on each other whereas low - coupled components are less dependent or almost independent on each other. Good design always have low coupling between components of system.

Example

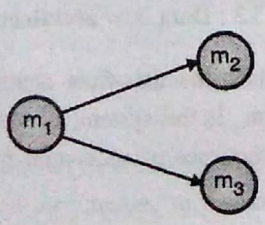


Fig. 3.1 : Coupling

Here module m_1 is dependent on modules m_2 and m_3 for its functionality i.e. if any one of m_2 and m_3 fails m_1 will also fail.

But m_2 and m_3 are free modules they do not depend on any other component for their functions. Thus success or failure of any other module does not affect modules m_2 and m_3. If modules are highly coupled with each other then change in any one of them force change in other modules or components. One of the solutions to reduce coupling between system components is a functional design. In general, in system development coupling is always constructed with the cohesion.

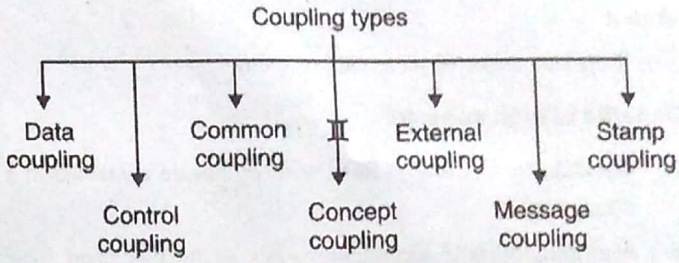


Fig. 3.2 : Coupling types

(i) Data coupling

Data coupling occurs between two components of system when they pass data to each other by parameters using argument list and each element in argument list is used. A component becomes difficult to maintain if too many parameters are passed.

(ii) Control coupling

When data are passed between two components and that affects internal logic of a component then there exists control coupling between those two components. It passes the control flags between system components.

(iii) Common coupling

Common coupling is also called as global coupling since in these coupling components of system communicates with each other by using global area. Common coupling becomes difficult to track when data in global area is changed.

(iv) Content coupling

When one component refers to the internals of the other component then those two components are said to be content coupled. As content coupling is worst type of coupling it should not be used in designing.

(v) External coupling

If two components of system share an externally imposed data format, communication protocol or device interface then they are said to be externally coupled. External coupling refers to communication between system components and external tools or devices.

(vi) Message coupling

Message coupling occurs between two components of system when those components communicate with each other via message passing.

Scanned by CamScanner

In this coupling components are independent of each other; thus it is low type coupling.

(vii) Stamp coupling

Stamp coupling occurs between two components of system when data between them are passed by arguments using a data structure containing elements which may or may not be used. Stamp coupling is also low type coupling.

Q. 7 What expected more : coupling or cohesion ? Why? May 17

Ans. :

Cohesion expected more than coupling

Benefits of high cohesion

1. **Readability :** (Closely) Related functions are contained in a single module.
2. **Maintainability :** Debugging tends to be contained in a single module.
3. **Reusability :** Classes that have concentrated functionalities are not polluted with useless functions.

Benefits of low coupling

1. **Maintainability :** Changes are confined in a single module.
2. **Testability :** Modules involved in unit testing can be limited to a minimum.
3. **Readability :** Classes that need to be analyzed are kept at a minimum.

Q. 8 Differentiate between cohesion and coupling.

Ans. :

Parameter	Cohesion	Coupling
Concept	Cohesion indicates relationship with in the module.	Coupling indicates the relationship between two or more different modules.
Represents	Cohesion represents relative functional strength of a module.	Coupling represents relative interdependence among the modules.
Degree	It is a degree up to which module focuses on systems' single function/component.	It is a degree to which one module depends on another module.
High or Low	High cohesion is good for system design.	Low coupling is good for system design.

Q. 9 Explain different Architectural styles. Dec. 16

Ans. : Architectural styles

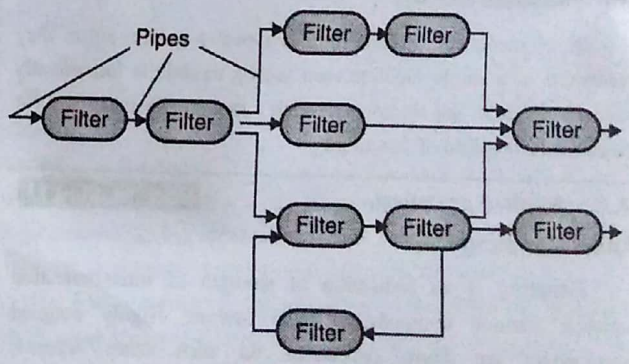
Architectural styles describe a set of systems which are internally linked with each other and share structural and semantic properties. Every architectural style defines a system type that includes the following :

1. Computational components like clients, server, filter, and database to execute the required system function.
2. A collection of connectors like procedure call, events broadcast, database protocols, and pipes to offers communication between the computational components.
3. Constraints to describe integration of components to form a system.
4. A semantic model, which allow the software designer so as to recognize the characteristics regarding the system as a whole by the way of studying the characteristics of its components.

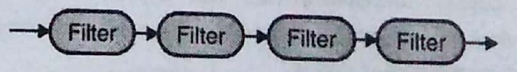
Following are the common architectural styles :

1. Data-flow Architecture

The purpose of data-flow architecture is to accept some inputs and transforms it into the required outputs by applying a sequence of transformations. Each component called as filter transforms the data and sends this transformed data to other filters for additional processing with the help of the connector called as pipe. Every filter operates as an independent entity, that is, it is not focused on the filter which is generating or consuming the data. A pipe is a unidirectional channel which transfers the data received on one end to the other end. It does not change the data in anyway; it just delivers the data to the filter on the receiver end.



(a) Pipes and filters



(b) Batch sequential

Fig. 3.3 : Data flow architecture

Most of the times, the data-flow architecture reproduce a batch sequential system. In this system, a batch of data is accepted as input and then a sequence of sequential filters are applied to transform this data.

Scanned by CamScanner

Example of this architecture is UNIX shell programs. In these programs, UNIX processes act as filters and the file system by which UNIX processes communicates act as pipes.

2. Object-oriented Architecture

In this architectural style, the components of a system wrap data and operations into single unit, which are used to change the data. In this style, components are represented as *objects* and they communicate with each other through methods (connectors).

3. Layered System Architecture

In layered architecture, every layer is responsible for performing a well-defined set of operations. These layers are organized in a hierarchical manner, where every layer is built upon one below another.

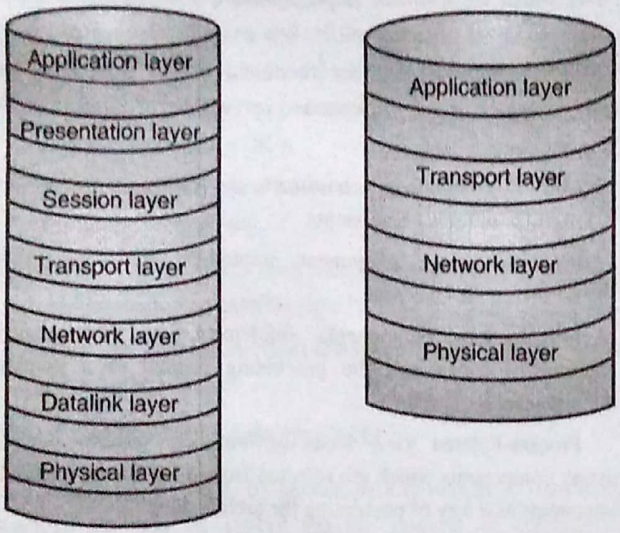


Fig. 3.4 : OSI and Internet Protocol Suite

Each layer offers a set of services to the layer which is located above it and acts as a client to the layer which is located below it. The interaction among layers is provided through protocols that describe a set of rules to be followed throughout the interaction. One common example of this architectural style is OSI-ISO (Open Systems Interconnection-International Organization for Standardization) communication system.

4. Data-centered Architecture

A data-centered architecture has two unique components :

1. Central data structure
2. Collection of client software

A central data structure is also known as data store. The data store such as database or a file demonstrates the current state of the data where as the client software performs various operations such as add, delete, update, etc. on the data which is stored in the data store.

Sometimes the data store permits the client software to access the data independent of any changes or the actions of other client software.

In this architectural style, new components related to clients can be added and existing components can be changed easily which does not require change in other clients. This is because client components work without depending upon any other components. A variation of this architectural style is blackboard system in which the data store is transformed into a blackboard that informs the client software when the data is modified. Additionally, the information can be transferred between the clients with the help of blackboard components.

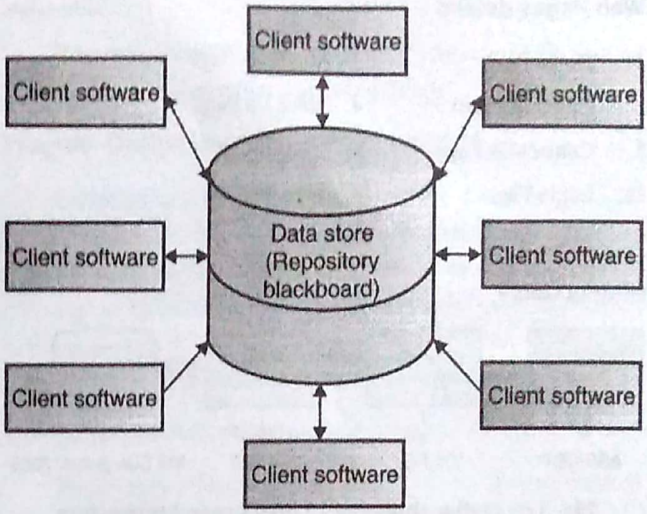


Fig. 3.5 : Data-centered architecture

5. Call and Return Architecture

A call and return architecture allows software designers to get such a program structure where modification is easy. This architecture style contains following two sub-styles.

Q. 10 Explain architectural design for e-commerce System. May 15, Dec. 15

Ans. : Architectural design for e-commerce

Description

Any member can register and view available products. Only registered member can purchase multiple products regardless of quantity. Contact Us page is available to contact Admin for queries. There are three roles available : Visitor, User and Admin.

1. Visitor can view available products.
2. User can view and purchase products.
3. An Admin has some extra privilege including all privileges of visitor and user.
4. Admin can add products, edit product information and remove product.
5. Admin can add user, edit user information and can remove user.
6. Admin can ship order to user based on order placed by sending confirmation mail.

Scanned by CamScanner

Using the code

Attach the database in your "SQL Server Management Studio Express". Run the application on Microsoft Visual Studio as web site. Locate the database.

Master Page details

Online Shopping Master Page (Similar Master Page for Visitor, User and Admin)

Web Pages details

1. Home Page
2. About Us Page
3. Clothing Page
4. Order Us Page
5. Contact Us Page
6. Admin Page
7. Login Page
8. Register Page
9. Track

Project detail

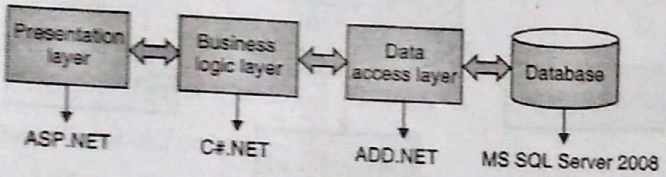


Fig. 3.6 : Online shopping - Three Layer Architecture

System requirements

Use-case diagram

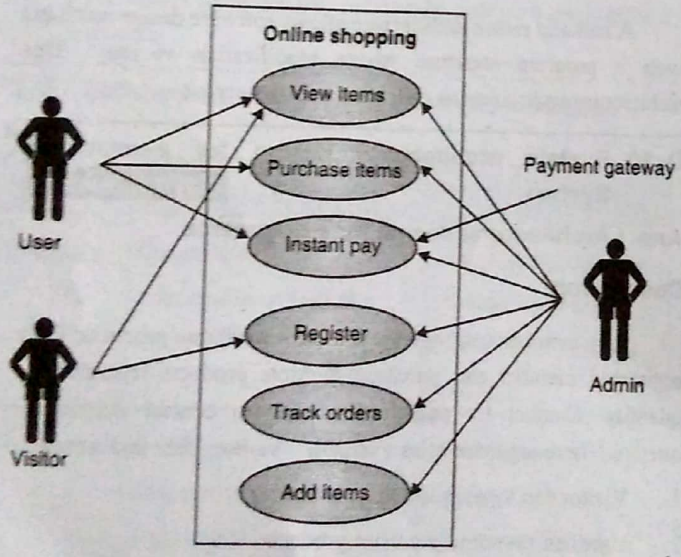


Fig. 3.7

Q. 11 Write note on Component Based Development. **May 16, Dec. 17**

Ans. : Component based development

The main function of component level design is to define data structures, algorithms, interface characteristics as well as communication mechanisms for all the present software components which are identified in the phase of architectural design. The phase of component level design is expected when the data, architectural, and interface designs are completed.

The software is represented by the component-level design in such a manner which lets the designer to review it for correctness as well as consistency prior to its built. The work product generated is considered as a design for all of the software components. It is represented with the help of graphical, tabular, or text-based notation. Design walkthroughs are carried out to observe accuracy of the data transformation or control transformation which has been allocated to all of the components in the earlier design steps.

Component definitions

A **component** is a modular, deployable, replaceable part of a system which encapsulates implementation and exposes a set of interfaces. As per object-oriented view a component contains a set of collaborating classes As per traditional view, a component (or module) resides in the software and serves one of three roles as follows :

1. Control components coordinate invocation of all other problem domain components
2. Problem domain components implement the functionality needed to the customer
3. Infrastructure components implement the functionality required to support the processing needed in a domain application.

Process-Related view focus on building systems out of existing components which are selected from a catalog of reusable components as a way of populating the architecture.

Component-level design guidelines

1. Components

Establish naming conventions in the phase of architectural modeling. Names of Architectural components should be meaningful for the stakeholders. Names of Infrastructure components must have implementation specific meanings. The nature of components can be identified with the help of UML stereotypes.

2. Interfaces

Use of lollipop representation is considered as better as compared to formal UML box and arrow notation. For the purpose of consistency, the flow of interfaces should be from the left-hand side of the component box. Display only those interfaces which are relevant to the component under construction.

3. Dependencies

To improve readability it should be better to have the model dependencies from left to right and inheritance from bottom (child classes) to top (parent classes). Interfaces must be used to represent component interdependencies rather than component to component dependencies.

Scanned by CamScanner

Conducting component level design

1. Identify all of the respective design classes which are corresponding to the problem domain.
2. Identify all of the respective design classes which are corresponding to the infrastructure domain.
3. Elaborate all of the design classes which are not acquired as reusable components.

(a) Specify details regarding the message when there is collaboration of classes or components.

(b) Identify suitable interfaces for all the components.

(c) Elaborate attributes and define data types and data structures which are necessary for their implementation.

(d) Describe processing flow of all the operations thoroughly.

4. Make the Identification of persistent data sources such as databases and files and recognize the classes which are necessary to manage them.
5. Develop and elaborate behavioral representations for all the classes or components.
6. Elaborate deployment diagrams for the purpose of providing additional implementation details.
7. Factor all of the component-level diagram representations and consider alternatives.

Object Constraint Language (OCL)

Complements UML by permitting a developer to use formal grammar as well as syntax for the purpose of constructing unambiguous statements regarding design model element.

Parts of OCL language statements :

1. A context which indicates the restricted situation in which the statement is considered as valid.
2. A property which represents few characteristics of the context.
3. An operation which manipulates or qualifies a property.
4. Keywords which are used for the purpose of specifying conditional expressions.

Conventional or Structured Component Design

Each block of code has a single entry at the top. Each block of code has a single exit at the bottom. Only three control structures are required : sequence, condition (if-then-else), and repetition (looping). Reduces program complexity by enhancing readability, testability, and maintainability.

Design Notation

Flowcharts : Arrows for flow of control, diamonds for decisions, rectangles for processes.

Decision table : Subsets of system conditions and actions are associated with each other to define the rules for processing inputs and events.

Program Design Language (PDL) : Structured English or pseudocode used to describe processing details.

Program Design Language Characteristics

Predetermined syntax with keywords provided for the purpose of representing all structured constructs, data declarations as well as module definitions. Free syntax of natural language for the purpose of describing processing features. Data declaration facilities for simple as well complex data structures. Subprogram definition as well as invocation facilities.

Design Notation Assessment Criteria

Modularity : Notation provide support for development of modular software.

Overall simplicity : Easy to learn, easy to use, easy to write.

Ease of editing : Easy to make changes in design representation whenever required.

Machine readability : Notation can be input directly into a computer-based development system.

Maintainability : Maintenance of the configuration in general engage maintenance of the procedural design representation.

Structure enforcement : Enforces the use of structured programming constructs.

Automatic processing : Allows the designer to verify the correctness and quality of the design.

Data representation : Ability to represent local as well as global data directly.

Logic verification : Automatic logic verification improves testing competence.

Easily converted to program source code (makes code generation quicker).

Chapter 4 : Quality and Configuration Management

What are the advantages of Software Testing ?

Ans. Advantages of software testing

1. Testing reduces the possibility of software failure by removing errors which leads to software failure.
2. Testing process removes maximum possible errors from software and helps to deliver qualitative software to customer.
3. Testing ensures correctness and completeness of software along with quality.

Scanned by CamScanner

4. Testing allows us to verify and validate the software and ensures that software will satisfy all of the user's requirements.
5. Testing enables the software to produce expected outcome by removing errors which prevent system from producing expected outcome.

Q. 2 Write note on Manual Testing.

Ans. :

Manual Testing

Manual Testing is one of the type of Software Testing in which Tester treat himself as end user and use every module of application from login module to log out module to check application behavior according to requirement provided by user. In manual testing, tester manually runs all the test cases without help of any automation testing tools. Manual Testing is the very basic type of all testing types and supports to search the bug (error) in the software system.

First we do manual testing for any new application before going for automation testing. We perform feasibility study of automation testing in manual testing i.e. do the analysis of benefits of performing automation testing for our project to take the decision that whether to do automation testing or not.

For performing manual testing, we do not need knowledge of any testing tool.

Manual Testing requires more effort, but is necessary to check automation feasibility. One important thing to know about Software Testing is that "**100% Automation is not possible**" so we need to perform manual testing.

Goals of Manual Testing

The main purpose of performing manual testing is to give assurance that our software product does not contain any defect and it fulfills all functional requirements of end user.

Test Suites (i.e. collection of test cases) or cases, are designed in the testing phase and they must test all functionalities present in our software product.

Manual testing gives assurance that reported defects are fixed by developer and tester performs retesting on it after fixing the defects. Mainly manual testing verifies the quality of the software product and deploys bug-free software to the customer.

List of manual testing is as follow :

(i) Unit testing (ii) Integration testing

(iii) System testing (iv) Acceptance testing

Q. 3 Differentiate between manual and automation testing.

Ans. :

Parameter	Manual Testing	Automation Testing
Human interaction	To perform manual testing human interaction is required for test execution.	To perform Automation Testing, human interaction is not required. In automation testing we use tools to run the test cases.
Requirements	To perform manual testing, we need skilled employees, more time and high costs.	To perform automation testing, we require automation tools. Automation testing saves time, cost and manpower. Once test suite is recorded in automation tool then it can easily run test suit.
Application to Test	Any application can be first tested manually. Informal testing like ad-hoc and monkey testing are performed manually.	Automated testing is used to test an application whose requirements are stable at some extent and automation testing mainly used to perform Regression Testing.
Execution of Same test cases	We cannot use manual testing while executing same test suit. Repetitive execution of test cases become boring and error prone.	The most boring part which is of running same test cases repetitively can be performed with the help of automation testing.

Q. 4 What are the objective of testing ?

Ans. : Objectives of testing

1. Finding defects which can be generated by the programmer while doing coding of software.
2. Report the detected defects to developer for correction and after correction retest the software product to give assurance that software does not contain any defect.
3. Software testing is done to give assurance that our software product fulfills all requirements of end user.
4. To give assurance that we deliver quality software to our end user.
5. To make sure that it satisfies requirements present in the BRS (Business Requirement Specification) and SRS (System Requirement Specifications).
6. To get the confidence of the customers by giving them a quality software product which must be user friendly and fulfills all the requirements and expectations of customer.

Scanned by CamScanner

Q. 5 Explain the software testing process in detail.

Ans. :

Software testing process

Software testing process is a sequence of various activities which helps to certify software system. Software testing process activities are : Requirement analysis, Test planning, Test case development, environment set up, Test execution, Test cycle closures. Each of the activity of software testing process has defined entry and exit criteria.

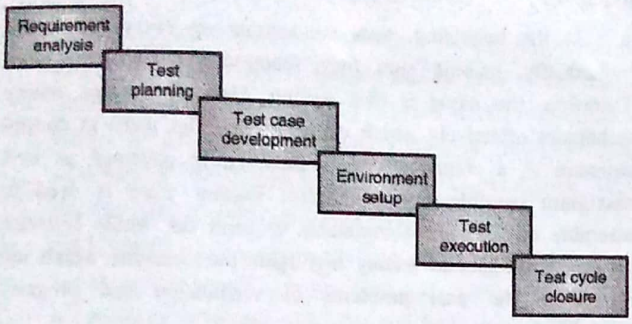


Fig. 4.1

1. **Entry criteria :** Entry criterions of testing are prerequisite conditions that must be satisfied before testing begins.
2. **Exit criteria :** An Exit criterion of testing describes the conditions that must be satisfied before testing is concluded.

Phases of Testing process

(i) **Requirement Analysis :** In this phase all gathered requirements of system (i.e. functional as well as non-functional) are studies to determine whether these requirements are testable or not. Depending on requirements, it is decided to perform type of testing, what should be focus and priority of testing. At the end of this phase Requirement Traceability Matrix (RTM) is prepared.

(ii) **Test planning :** It is also known as test strategy phase. It is responsibility of test manger. In this phase manager finds out required efforts and estimated cost for the system under development. At the end test plan is prepared as an outcome of test planning.

(iii) **Test Case Development :** In this phase all test data is determined, reviewed and networked if necessary. It includes generation, verification and rework (as per requirement) of test cases and test scripts. Test cases and test scripts are outcome of this phase.

(iv) **Test environment set up :** After completion of set up, smoke test is performed on it. Thus ready test environment and result of smoke test are outputs of this phase.

(v) **Test Execution :** During this phase testing of software is carried out by test team using previously prepared test plan and test cases. Intension of test execution is to find bugs in software. These bugs are reported to development team for correcting them. After correction retest is performed in order to assure that reported bugs are removed.

(vi) **Test cycle closure :** This is end of software testing process. In this phase cycle completion criteria is determined depending on time, test coverage, quality of system, cost of software, business objectives etc.

By using all these attributes test metric is prepared. Before this phase testing has been completed and defect logs are available. Using or analyzing defect logs, test closure report is prepared.

Q. 6 Compare Verification and Validation Testing. May 17

Ans. :

Parameter	Verification	Validation
Focus	Concentrates on right way to build a system.	Concentrates on output of build system.
Definition	Verification can be defined as it is a process of estimating the intermediary work products of a software development lifecycle to verify that we are in the correct track of creating the end user product.	Validation can be defined as it is the process of evaluating the final product to ensure that the software meets all the specified requirements.
Aim	The aim of Verification is to ensure that the product being develop is as per the requirements and design specifications.	The aim of Validation is to ensure that the product really meets up the user's requirements, and verify whether the specifications be correct in the first place.
Inclusion	Verification contains Reviews, Meetings and Inspections.	Validation contains testing such as black box testing, white box testing, gray box testing etc.
Performer	Verification is conducted by quality assurance team.	Validation is conducted by testing team.
Execution of code	Execution of code is not a part of Verification.	Execution of code is a part of Validation.

Scanned by CamScanner

Parameter	Verification	Validation
Explanation	Verification process gives explanation about whether the outputs are as per the inputs or not.	Validation process gives explanation about acceptance of software by the user or not.
Processes	Plans, Requirement Specifications, Design Specifications, Code, and Test Cases etc. are evaluated in verification.	Testing of actual software is done in validation.

Q. 7 Explain "Testing Strategy". **Dec. 16**

Ans. :

Testing Stratergy

The software process can be considered as the spiral as shown in Fig. 4.2.

In the beginning, the role of software is defined by the system engineering and proceed to software requirements analysis, in which there is establishment of the information domain, function, behavior, performance, constraints, and validation criteria for software. Moving towards inward in the spiral, we come to design and at the end to coding.

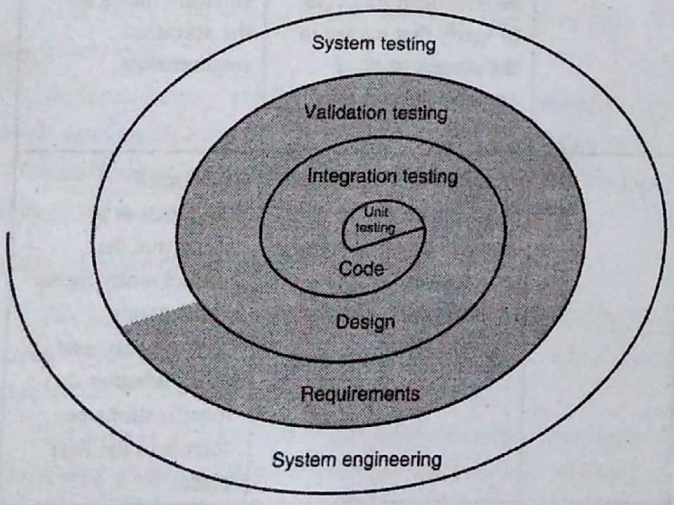


Fig. 4.2 : Software Process

It is possible to view a strategy for software testing in the context of the spiral. Unit testing gets started at the vortex of the spiral and focus on all of the software units such as component, class, or WebApp content object which are implemented in source code. Testing continues by traversing outward direction along the spiral to integration testing. Here the main concentration is on designing and building the software architecture.

Taking next turn outward on the spiral, we can observe validation testing, in which requirements which are set as part of requirements modeling are validated against the respective software which has been developed. At last we reach at system testing, where testing of the software as well as other system elements is done as a whole. For testing any computer software, we have to spiral out in a clockwise direction along streamlines which enhances the scope of testing with each and every turn.

Taking into account the process from a procedural point of view, in the context of software engineering, practically testing is an implementation of four steps in a sequence. The steps are shown in Fig. 4.3

In the beginning, tests concentrate on every component individually, making sure they function properly as a unit. Therefore, the name is unit testing. Unit testing use testing techniques effectively which exercise particular paths in control structure of a component to confirm entire coverage as well maximum possible error detection. Further there is need to assemble or integrate components to form the whole software package. Integration testing highlights the concerns which are related to the dual problems of verification and program construction.

Test case design techniques which in general used to concentrate on inputs and outputs are more common in the process of integration.

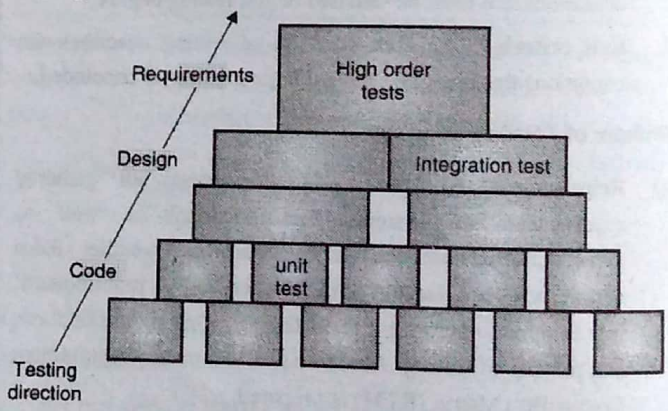


Fig. 4.3: Four Steps of Testing

After integration of software, a series of high-order tests is conducted. There is need to evaluate validation criteria which has been set during requirements analysis. Validation testing presents absolute assurance that all informational, functional, behavioral, and performance requirements are fulfilled by the software.

The final high-order testing step is considered outside the limit of software engineering and in the wider context of computer system engineering. It is necessary to combine the software which is validated with other system elements such as hardware, people, database, etc.

System testing confirms that all the elements mesh correctly and the overall system function/performance has been achieved.

Q. 8 Draw and explain stub and driver mechanism of unit testing.

Scanned by CamScanner

Ans. : Stub and driver

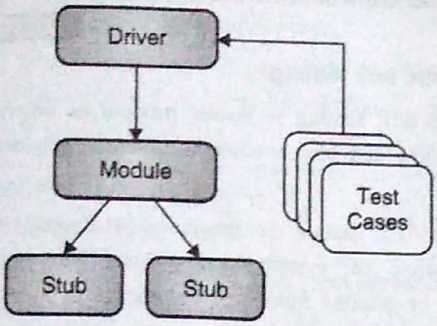


Fig. 4.4 : Stubs and Drivers

1) **STUBS :** Assume we have 3 modules, Module A, Module B and Module C. Module A is ready and we need to test it, but module A calls functions from Module B and C which are not ready, so developer will write a dummy module which simulates B and C and returns values to module A. This dummy module code is known as **stub**.

2) **DRIVERS :** Now suppose we have Modules B and C ready but Module A which calls functions from Module B and C is not ready so developer will write a dummy piece of code for Module A which will return values to Module B and C. This dummy piece of code is known as **driver**.

Q. 9 Differentiate between Alpha and Beta Testing. **May 17**

Ans. :

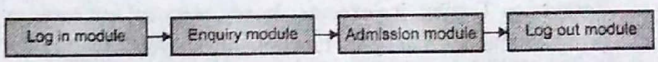
Parameter	Alpha Testing	Beta Testing
Performed by	Alpha testing is done by Testers who are generally employees of organization.	Beta testing is done by End Users who are not employees of organization.
Testing environment	Alpha Testing done is in lab environment.	Beta testing is done in real world environment.
Factors tested	Security Testing and Reliability are not tested in depth during Alpha Testing.	Reliability, Security, Robustness is verified during Beta Testing.
Included testing type	Alpha testing includes white box and black box testing.	Beta Testing includes Black Box Testing.
Time required	Alpha testing requires lot of time to perform because in it white box as well as black box testing are performed.	Few weeks of time is required to perform the beta testing.
Fixing of issues	Important issues can be corrected by developers without delay in Alpha testing.	Issues detected in Beta testing will be corrected in future versions of the software product.
Focus	Alpha testing focuses on giving assurance about quality of the software product before going to Beta testing.	Beta testing also focus on quality of the product, but it collects feedback from end users about software product and gives assurance that the software product is ready for use in real world environment.

Q. 10 Explain the need for system testing. **May 15**

Ans. : Need of system testing

System Testing is testing of entire and completely integrated software product. System Testing is sequence of various tests whose work is to test the complete computer based system. System testing is end-to-end testing. i.e. we test system from log in module to log out module.

For e.g. In college admission project, system testing carried out as :



System testing contains both functional (check whether requirements of user are fulfilled or not) as well as Non-Functional testing (check whether expectations of user are fulfilled or not).

There are Two basic categories of Software Testing such as

1. **Black Box Testing :** In this type of testing, we check working of our software project by running it. We do not need to check code of our software in this testing type. **System testing** is included in black box testing
2. **White Box Testing :** In white box testing we check internal working of our code. So in this type of testing tester need deep knowledge of programming language which is used in our software product.

Types of System Testing

1. **Usability Testing :** Usability Testing is a type of software testing in which a group of end-users of a software system use software to check user friendliness. It is non-functional testing. This testing basically concentrates on how easily user handles the application. It is verified that after few hours training, end user can use system comfortably. Icons used in user interface are not confusing. e.g. for cut option we use scissor picture, not CD picture. This type of testing is also known as User Experience Testing. This testing is suggested to perform at the initial design phase of SDLC because it find out bugs (errors) in user interface and give chance to improve them.
2. **Load Testing :** Load testing is a type of Performance Testing which specifies performance of system under real-life load conditions. Load testing helps to specify how the application behaves when multiple users access it at same time.

Scanned by CamScanner

This testing normally used to find out :

1. The maximum number of users who can access the software at same time.
2. Specifies whether currently available infrastructure i.e. software and hardware is adequate to execute the application.
3. Check what happen when maximum number of users accesses the system simultaneously.
4. Scalability (action taken to increase capacity of system such as increase storage etc.) to permit to more number of users to access our application.

It is non-functional type testing. Load testing is normally used when we test Client/Server based applications and Web based applications.

3. **Regression Testing :** Regression Testing is a type of software testing which is used to check whether changes have been made in code due to some error or change in requirement affects existing working functionality. In Regression Testing, we execute already executed test cases to give assurance that old functionalities work well after performing changes in code. This testing is performed to give guarantee that new code added in our software does not disturb working of existing functionalities.
4. **Recovery Testing :** Recovery Testing is done to specify whether system recovers itself after the system crash due to disaster such as power failure or network not available etc. In recovery testing, system perform rollback i.e. identify the point where the behavior of the system was correct and then from that point again perform the operations up to the point where system get crashed.
5. **Migration Testing :** Migration testing is performed to give assurance that the software can be moved from older system infrastructures to current system infrastructure without any problem. In Migration testing, we check data from old system for the compatibility with new system.
6. **Functional Testing :** Functional testing is a type of software testing which checks that every **function** present in the software application works as per requirements of user. This testing includes black box testing and it does not focus on the source code of the software application. Every functionality of the system is verified by tester using appropriate test data and actual result is compared with expected result. If there is difference between actual result and expected result then bug is considered. Functional testing is done using Requirement Specification Document.
7. **Hardware/Software Testing :** In Hardware/Software Testing, we perform testing of communication between the hardware and software used in our system. IBM called the Hardware/Software Testing as "HW/SW Testing".

Q. 11 Describe in detail white box techniques. May 15, Dec. 15

Ans. : White box testing

White Box Testing is testing process in which we verify internal coding and infrastructure of software product under the test.

White box testing is based on analysing the internal implementation of system under test. Thus programming knowledge or detailed functional knowledge of system is major pre-requisite for tester. White box testing mainly concentrates on the testing flow of inputs and outputs through the software, strengthening security and improving design and usability of software product. White box testing is also called as Clear Box testing, Open Box testing, Structural testing, Transparent Box testing, Code-Based testing, and Glass Box testing. The term "white box" is used to indicate testing of code of software because we can see what is present inside the box.

The clear box or white box term indicate the capability to see via outer part of software into its internal workings.

It uses following methods :

1. **Statement coverage :** i.e. testing all programming statements using minimum number of tests.
2. **Branch coverage :** This is to ensure that all branch conditions in system are tested at least once.
3. **Path coverage :** It includes ensuring that each statement and branch in system is tested at least once.

Other than above there are different coverage types such as Condition Coverage, Multiple Condition Coverage, Path Coverage, Function Coverage etc.

Every technique has its own advantages and tries to cover all parts of software's product code. By using Statement and Branch coverage, we can perform 80-90% code coverage which is enough.

White box testing involves two steps. They are :

Step 1 : Understand the source code

In this step, testers understand the source code of the software product. White box testing includes testing of internal working of software product. So the tester must have very high level knowledge of programming languages used in the software product under the test.

Security is basic purpose of performing the software testing. So the tester should be able detect security problems and protect the software from attacks by hackers and naive users who may add malicious code in software product knowingly or unknowingly.

Step 2 : Create test cases and execute

The second step involved in white box testing includes testing the source code of software product under the test for checking of proper flow of control and structure. The tester will generate small test cases for every process or group of processes in

Scanned by CamScanner

the software product. This method requires that the tester should have high level knowledge about source code.

White box testing may be performed by the developers because developers have high level knowledge of programming language used in software product under the test.

Q. 12 What is Black Box testing ? Dec. 17

Ans. : Black box testing

The software testing methods which test the functionality of an application without knowing its internal structure, coding information and knowledge of internal paths of the software is called black box testing. In this method test cases are built based on what application is supposed to do. It is also known as behavioural testing or specification based testing.

In Black Box Testing we concentrate on inputs provided to software product and output from the software product without applying compulsion about knowledge of the programming language used in software under the test. Black box testing can be applied during each level of software testing process.

It makes use of various testing techniques like Boundary Value Analysis (BVA), equivalence class decision models etc.

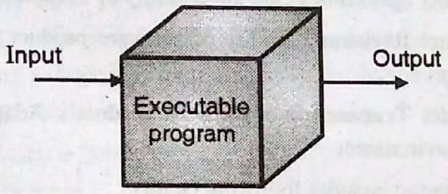


Fig. 4.5 : Black box testing

Following are the steps which are used to perform black box testing :

1. First the requirements and specifications of the system are analyzed.
2. Tester select valid test data i.e. data for positive test scenario to verify whether application under test is able to processes that data and give output as expected.
3. Also tester selects invalid test data for negative test scenario to verify whether application under test is able to processes that data and give output as expected.
4. Tester state expected outputs for each test data.
5. Software tester generates test cases with the selected test data. Then test cases are run.
6. Software tester performs comparison between the actual output and expected output.
7. Defect is detected if actual result is not same as expected result.
8. Then detected defect is reported to developer.
9. Developer can then fix that defect and then application is retested by tester.

For developing test cases following methods are used in Black Box Testing :

(i) **Boundary Value Analysis (BVA) :** As the name indicates boundary value analysis is a process of testing boundaries of the input values. It is most commonly used technique in BBT. Here basic idea is to select input values at their; minimum, just above minimum, normal value, maximum value and just below maximum value. BVA always comes after the equivalence class partitioning.

(ii) **Equivalence class partitioning** : Equivalence class partitioning reduces the number of all possible inputs by dividing them into classes. It tests application thoroughly and avoids redundancy of input values. It can be applied at all levels of testing. It always performed before boundary value analysis technique is applied. BVA and equivalence partitioning are closely related and used together.

(iii) **Cause - effect graphing : This** technique graphically explains relationship between output of system and all the factors that affect the output of system. It helps to identify all possible causes of any specific effect. This technique tests only the external behaviour of system. It makes selection of test cases such that they logically relate cause to the effects.

(iv) **Error – Guessing :** Error guessing is a technique which makes use of tester's experience and skill for testing similar applications to find out defects which may not be determined by formal techniques. It is usually done after formal testing techniques are applied. It list outs all possible defects in system and then design tests which will attempt to produce them. The success of error guessing technique is mainly depends on knowledge and skills of tester.

Types of Black Box Testing

1. **Functional testing :** Functional testing is a type of software testing which checks that every function present in our software application works as per requirements of user or not.
2. **Non-functional testing :** Non-functional testing is used to check non-functional requirements such as performance, scalability, usability, and security are fulfilled or not.
3. **Regression testing :** Regression Testing is used to check whether changes which have been made in code because of some errors or change in requirement affects existing working functionality.

Q. 13 Compare white box testing and black box testing. May 17

Scanned by CamScanner

Ans. :

Parameter	Black box testing	White box testing
Applicable levels	It is mainly applicable at higher levels of testing : Acceptance and system testing.	It is mainly applicable at lower levels of testing i.e. unit and integration testing.
Base for testing	Requirement specification is the basis for Black Box Testing.	Detail design is the basis for White Box Testing.
Performed by	It is carried out by software testers	It is carried out by software developers.
Aim	It aims at what functionality is system performing.	It aims at how system is performing its functionality.
Requirements	Programming knowledge and implementation knowledge is not necessary to carry out BBT.	Programming knowledge and implementation knowledge is necessary to carry out WBT.
Cost	It is less expensive than WBT.	It is more expressive than BBT
Focus on	In black box testing, we do not test the code and focuses on the functionality of software product.	In white box testing, we test the code and focuses on the security of software product and flow of control.
Facility provided	Black box testing provides facility of verifying communication among different modules present in software product.	White box testing does not provide facility of verifying communication among different modules present in software product.

Q. 14 What is quality? Explain McCall's quality factors.

May 16, Dec. 16, May 17, Dec. 17

Ans. : Quality

The main goal of software engineering is to develop high quality software. Quality software is the one which is defect or bug free, delivered on time, cost effective, meets all requirements and expectations, and is maintainable. Software quality based on the various factors which will vary as the customer requirement changes or use of that software varies across different applications.

McCall, Richards, and Walters propose a useful characterization of factors that affect the quality of software.

Following are the factors / **attributes** that affect the quality of the software :

1. Directly measurable factors
2. Indirectly measurable factors

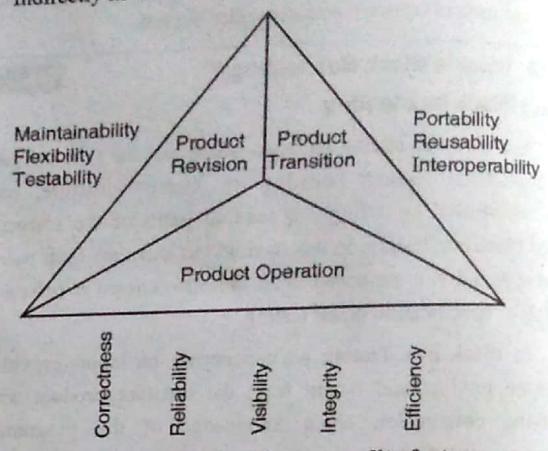


Fig. 4.6 : McCall's software quality factors

As shown in the Fig. 4.6 the focus is on three important aspect of software :

1. **Product operation :** Operation ability of Software product.
2. **Product Revision :** Ability of software product to undergo change.
3. **Product Transaction :** Software product's Adaptability to new environment

Each group includes following factors :

(a) Product Operation

(i) **Correctness :** If software product provides required functionality properly as specified in software requirements then the software product is correct. Software product must satisfy the customer objective.

(ii) **Reliability :** Number of errors in the software product should be kept as minimum as possible, because when amount of errors is less, then the reliability of the system will be more.

(iii) **Usability :** Software product must be user friendly, i.e. easy to learn, use, input data, interprets the output, and recover from errors.

(iv) **Integrity :** Software product should provide integrity of the data. That means the data must be protected from unauthorized access.

(v) **Efficiency :** With fewer amounts of time and resources the software product should function in proper way.

(b) Product Revision

(i) **Maintainability :** It is related to maintenance phase of software development. Software product maintenance should require less effort to recover from an error.

(ii) **Flexibility :** Software product should adapt any changes occurred in the system. It means that software product should be flexible in nature.

Scanned by CamScanner

(iii) **Testability :** Testing the software product is important to ensure that the software product performs its specified functions.

(c) **Product Transaction**

(i) **Portability :** Efforts needed to transfer software product from one machine to other machine and operating systems.

(ii) **Reusability :** The ability to reuse the program or part of program in another application.

(iii) **Interoperability :** Ability of Interacting the software with another system.

Q. 15 Explain change control activities in SCM.

May 15, Dec. 15, Dec. 16, May 18

Ans. : Change control activities

Changes occur at all phases in the software lifecycle. Design or implementation changes may be necessary if requirements changes, or when deficiencies are identified in the design or implementation approach to a stable requirement. Testing may uncover defects that require changes in the code or the design and requirements.

Changes must be made to the right version of the code, testing must verify performance of the change and the integrity of the remaining software, and all associated documentation must be updated to be consistent with the final code.

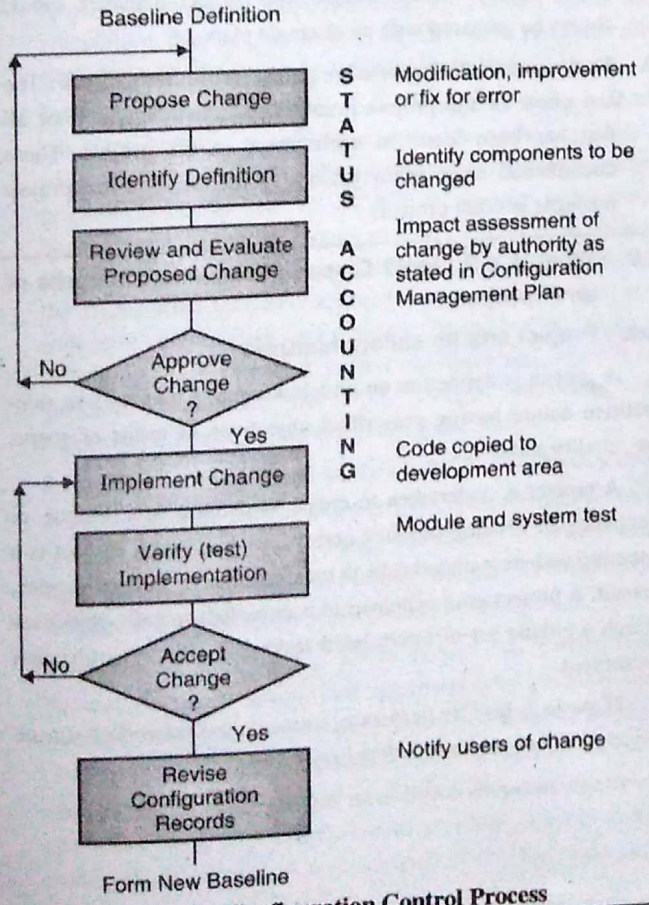


Fig. 4.7 : Configuration Control Process

A mechanism is needed to process change requests (e.g., discrepancies, failure reports, requests for new features) from a variety of sources throughout the development process and during operation and support of the software product. This mechanism should extend to a definition of a process to track, evaluate, and implement change requests into a new version and new release of the software. Generally, no single procedure can meet the needs of all levels of change management and approval levels. The minimum activities needed for processing changes include :

Defining the information needed for approving the requested change. Identifying the review process and routing of information. Describing the control of the libraries used to process the change. Developing a procedure for implementing each change in the code, the documentation, and the released software product.

Q. 16 Explain version control activities in SCM.

May 15, Dec. 15, May 18

Ans. : Version control activities

Version control integrates the procedures with tools for the purpose of managing different versions regarding the configuration objects which are generated throughout the software process.

Following are the important capabilities which a version control system implements or is straightforwardly integrated with :

1. A project database (repository) which holds all the appropriate configuration objects,
2. A version management capability which holds all the relevant versions of a configuration object,
3. A make facility which helps to get all respective configuration objects and build a particular version of the software.

Additionally, version control as well as change control systems usually implement an issues/bugs tracking capability which helps the team to record and track the status regarding all of the important issues related with each and every configuration object. Most of the version control systems create a change set of all the respective changes which are necessary to build a particular version of the software. It is possible to identify various named change sets for an application or system. This helps in building a version of the software by mentioning the change sets (by name) which must be applied to the baseline configuration.

A system modeling approach is used to accomplish it. The system model contains:

(1) A template which contains a component hierarchy and a "build order" for those components which gives information regarding how the system must be constructed,

(2) Construction rules, and

(3) Verification rules.

Now-a-days there are several automated approaches have been proposed to version control.

The basic distinction in those approaches is the sophistication of the attributes which helps in building specific versions.

Scanned by CamScanner

Q. 1 Why is an increased need felt for effective and efficient project management?

Ans. :

Few reasons why an increased need is felt for effective and efficient project management:

1. **To control scope of project and manage change** : Although the project deliverables are defined at the outset of the project, small changes in project deliverables are common. These changes are demanded by customers, stakeholders, management, suppliers or the project team itself. Individually, these change demands may seem acceptable and manageable, but collectively these change demands can lead to a significant expansion in the project scope and can lead to an overrun in schedule and budget. However, with project management, if the manager effectively manages the scope of their project, they have a better chance of effectively managing project resources and change.
2. **To deliver projects on time and within budget :** The project management process includes cost calculations such as return on investments (ROI). Once ROI is established it is for the project manager to ensure that the project schedule and budget are adhered to else the project will fail to deliver the expected results.
3. **To ensure the focus of the project team :** It is common for the project team to drift from the main tasks and spend unnecessary longer time on other tasks. Hence, it is the responsibility of the project manager to ensure that the project team focuses on the right tasks by using a clear and concise project charter and that there are no interferences.
4. **To collect user requirement from disparate sources :** The project manager at the initiation phase should collect user requirement, project constraints and conduct a feasibility study to build a strong business case justification. The primary advantage of collecting input from various sources is that the project manager is able to avoid future dissent from users and is able to communicate project benefits.
5. **To define the critical path to optimally complete the project :** Every project is made up of connected activities each having their individual constraints. By using the critical path method technique the project manager is able to identify the critical path and thus ensure the successful completion of the project.
6. **To provide a process for estimating project resources, time and costs :** Solid project management tools and techniques and past experience will enable the project manager to correctly estimate the project resources requirement as well as determine the time required for the completion of the project and the likely expenditure. Such, estimation at the initiation phase will ensure that the project receives adequate commitment from the management and its success is ensured.
7. **To communicate project progress, risks and changes :** The stakeholders of the project need to be kept updated on the project progress, hurdles encountered and changes incorporated. Every good project management plan has a communication plan that addresses communication issues, provides formats and lays process for execution of the communication plan.
8. **To explore project assumptions :** Although ample ground work is done for a project there are bound to be a few assumptions on which the project is based. Hence, a good project manager has to delve deeper into user requirements, project constraints and management expectations to understand the hidden project requirements. A project based on too many assumptions could ruin its chances of success.
9. **To prepare for unexpected project issues :** Howsoever, one may be prepared there are bound to be a few issues which may suddenly surface. Hence, the project manager should always be prepared with an alternate plan.
10. **To document the knowledge gained from the project :** The last phase of the project involves the documentation of all that has been learnt at each phase in the project. These documented experiences provide guidance to other project manager in other projects.

Q. 2 What is a Project? Discuss the salient features of an IT project.

Ans. : Project and its salient features

A project is defined as an undertaking of a non-routine, non-repetitive nature having prescribed objectives in terms of scope, time, quality and cost.

A project is undertaken to create something new, unique, or to enhance an existing product, service, or system. A project is a temporary endeavor undertaken to create a unique product, service, or result. A project is an endeavor to accomplish specific objectives through a unique set of interrelated tasks and effective utilization of resources.

IT projects involve hardware, software and networks to create product, service, or result. A few examples of IT projects are;

1. A new feature is added to an existing software application.

Scanned by CamScanner

2. New system is developed to increase sales force productivity, improve customer relationship management and enhance supply chain management.
3. A firm decides to implement an integrated ERP project to consolidate its information systems.
4. Replacing the company's manual time keeping system with a web based system within a particular time frame.

Q. 3 State the attributes of a project.

Ans. : Project attributes

1. **Purpose :** A project has a unique purpose and well defined objectives. It begins with rough ideas and an initial plan which is updated as more information emerges. Projects are undertaken to accomplish something of value to the company, maybe, a system or software. The sole purpose of undertaking a project is to produce a tangible product of value to the company.
2. **Goal :** A goal drives a project. It is the sole motivating factor and defines each activity, task, work, schedule and budget of the project. It provides direction to the team. Hence, the project goal should be clearly defined, ambiguity in defining the project goal leads to a project with no end.
3. **Time frame :** Since a project is a temporary endeavour it has to have a definitive start and end. The time frame for achieving the project goal is estimated based on the duration of the various project activities. The completion date of the project is set accordingly. However, for projects where the completion date is fixed, such as the Y2K problem, the starting date has to be set by working backwards.
4. **Interrelated tasks and resources :** A project is composed of interrelated tasks and utilizes resources such as people, software, hardware and other assets of the company. The primary asset for an IT project is people who cross departmental and other boundaries to achieve this unique purpose. Most projects demand the service of people outside the organization such as consultants. Resources, however, are expensive and limited hence should be used effectively to meet project objectives. A project is unique and hence it is difficult to estimate the time, budget and resources requirement. Hence, a project involves uncertainty.
5. **Ownership :** A project has various stakeholders but only one primary customer known as the project sponsor. The project sponsor is the "go to" entity for funds, direction and approvals regarding all project related matters. For the project manager the project sponsor is the owner of the project.
6. **Risks and Assumptions :** Every project has certain risk element, the more complex and big the project greater is the risk. Risk is directly proportional to the size and complexity of the project. Risks can emanate from sources both internal and external to the project team. Internal risks could be the resignation of a key project member, while external risks could arise from vendors and suppliers of key hardware and software. Assumptions are those unknown variables which are used to estimate key facets of the project. Therefore, the project manager has to identify all the risks and assumptions that could impact the project.

Q. 4 Enumerate the 4P's of project management.

Ans. : 4P's

Similar to any other engineering project the primary focus for effective IT project management are the 4P's namely people, product, process and project.

People

In software project management the people's component is probably the most crucial as they form the backbone of the project.

Although, other resources such as materials, facilities, tools and equipments are also of vital importance, the primary resource in IT projects is its people. IT projects are completed by people and the project manager will need the right people to accomplish project objectives. The complexity of modern IT projects makes it necessary to employ the services of individuals with varying skills and knowledge. Following are the key players in an IT project.

1. **Project manager :** The project manager is the captain of the team and is responsible for the success or failure of the project. The project manager has to ensure that all development requirements and processes are in place and are carried out as per the specifications and quality parameters specified.
2. **Project sponsor :** The project sponsor could be the client or an organizational appointee who is the "go to" person for all project related resource requirement and direction whenever required.
3. **Subject Matter Expert (SME) :** It is not necessary for the project manager and team members to be experts in the core subject of the project. Hence, every project needs a subject matter expert (SME) who will support the project by providing vital inputs on the subject matter. The SME could be user, client, or a subject expert who has specifically been roped in for the project.
4. **Project team member** - The project team is formed according to the requirement of the project and could comprise of analyst, programmers, network specialists, graphic designers, etc.

Product

This is the first step in project conceptualization and should be undertaken even before the project is planned. The product is the output of the project and hence its objectives and scope should be defined, alternative solutions should be considered and project constraints in terms of technical and management issues should be identified. Once this has been done it is possible to estimate cost of the project with reasonable accuracy, effectively assess risks to the

Scanned by CamScanner

project, break the project into tasks and manage the project schedule.

The objectives of the project identify the overall goals for the product without considering how they will be attained. The product scope identifies the primary data, functions and behaviors that characterize the product. Once the product objectives and scope have been understood and alternative solutions have been properly evaluated then a solution can be arrived at.

Process

Project Process is the application of knowledge, skill, tools and techniques to project activities to meet project requirements. Project management is accomplished through processes, which support all the activities necessary to implement the project, using project management skills, tools and techniques that receive inputs and deliver outputs.

In order for the project to be successful the project manager should select appropriate processes within the process groups that are required to meet the project objectives. The project manager has to adapt the product specifications and plans to meet the requirements of the project and product.

A process is a set of interrelated actions and activities that are performed to achieve a pre-specified set of products. The project management processes are undertaken by the project team and fall under one of the following category.

(i) The project management processes are common to most of the projects and are associated with each other by their performance for a common purpose. The common purpose is to initiate, plan, execute, monitor, control, and close a project.

(ii) Product oriented processes specify and create the project's products. Product oriented processes are defined by the project lifecycle and vary by application areas. Project management processes and product management processes interact throughout the project.

Project management processes focus on defining, coordinating and controlling the activities needed to manage the project while product management processes focus on the final product i.e. the tangible output of the project, namely, the information system.

Unlike, project management processes product management processes necessitate specific domain knowledge to complete the work.

Thus, every project will require a subject matter expert who has the requisite knowledge, tools and techniques to handle the product processes.

However, there must be a balance between project management processes and product management processes. For successful project management the project management and product management need to be properly aligned and connected with each other's processes to facilitate coordination.

Balance is the key, a solo focus on project management processes will not enable the project team to define the project scope or develop a quality system and on the other hand a solo focus on product management processes will not ensure the completion of work as required.

Project

The project manager as a key person is expected to carry out various tasks and lead the project team. The role of a project manager involves handling all aspects of the project such as planning, brainstorming and overseeing the overall completion of the project. Therefore, for a project manager the trick for successful project management is to focus on the five most important goals associated with project management. Meeting these five goals of a project will ensure project success. Project goals keep the focus on what is important for the success of the project.

The five most important goals of every project are

(i) Ensure that the project is completed within the deadlines set

(ii) Ensure that the project is completed within budget

(iii) Ensure user satisfaction

(iv) Ensure that all user requirements are duly met

(v) Ensure team management

Q. 5 Briefly describe the various phases of a generic project lifecycle.

Ans. : Phases of a generic project lifecycle

Projects do not form the core activity of the organization they are temporary activities with a definitive objective. Today, the project lifecycle approach is used in construction, aerospace, government agencies, research and development, manufacturing, electronics and many other industries; however the term project lifecycle may mean different things to different people depending upon the particular application.

The basic project lifecycle principles are applicable to a wide variety of projects such as new applications, major maintenance and repair projects, relocations and re-organizations, creating new web site, or establishing a new data base centre. Irrespective of the type of project the basic framework shared by every project remains the same.

The project lifecycle describes the various logical phases in the life of a project from its beginning to its end in order to deliver the final product of the project. The idea of breaking the project into phases is to ensure that the project becomes manageable, activities are arranged in a logical sequence, and risk is reduced. The phases are so created that each phase at its end provides one or more deliverables. Deliverables are tangible and verifiable products of work that are produced at the end of each phase and define the resources needed for the completion of each phase.

Scanned by CamScanner

The review of the deliverables at the end of each phase enables the project manager to evaluate the performance at the end of each phase and take necessary corrective action if felt necessary.

Although, it is recommended to undertake the phases in a linear sequence, phases could be overlapped to save time. However, overlapping of resources is risky and should be justified.

1. **The Project Goal :** The first step for any project, irrespective of its size and complexity, is defining its overall goal. Every project undertaken aims to provide business value to the organization hence the goal of the project should focus on doing the same. Now, the goal of the project needs to be defined initially and clearer as it provides the project team with a clear focus and guides it through each phase of the project. The project is hazy and seems risky at the start, but as the project goal gets defined and starts making progress, things start to look brighter and the probability of success increase. At the beginning of the project, the requirement of resources in terms of staff and cost is low; however, it increases as the project makes progress and reduces at the completion stage of the project.

2. **The Project Plan :** The project's goal provides the guidelines and makes it easier for the development of the project plan also known as baseline plan. The project plan is developed to provide answers to various project related queries such as;

   What the project aims to achieve? – The project deliverables

   How does the project team aim to achieve it? – The tasks and activities

   Who all will be involved in the project? – The stakeholders and the project team

   How much will it cost? – The project budget

   How much time will it take? – The project schedule

   What are the risks involved? – Risk identification

   The initial plan serves as a base to gauge the progress made by the project throughout its lifecycle.

3. **Project Plan Execution :** The project plan thus developed needs to now be executed. The project's progress should be compared with the baseline plan and communicated to all of the project's stakeholders. The project manager has to constantly monitor the projects' progress, schedule, budget, and resources.

4. **Project Closure :** Project closure marks the culmination of the project; the project may have completed all the work as planned or the project sponsor may have called of the project due to various reasons which we shall study later on in this book. However, in the event of the project having completed all the work as planned, the project sponsor should formally acknowledge its successful completion, discharge all resources from the ongoing project, and make necessary payment of dues of any external consultants and suppliers. The project closure is also marked by the presentation of a project report that documents all the project deliverables.

5. **Project Evaluation :** It is not possible to evaluate the true value of any IT project immediately after its implementation. Sometimes the goals are long term and the results of the project can only be gauged after the passage of a certain period of time. Merely evaluating the success achieved in implementing the hardware and software peripherals of the project does not amount to having succeeded in the project. These are merely means to an end and hence time only will tell whether the project has succeeded in accomplishing what it was set in motion for. However, this does not mean that evaluation should not be undertaken immediately. For starters, the project team should document all its experiences - things it did right, things it should have done differently, and all the lessons learned during the entire course of the project. This document proves useful in the execution of future projects for the organization. Also, the project manager and his team should be evaluated for their performance. While, the project manager should evaluate each team member and provide feedback to the member as well as the management, the performance of the project manager should be evaluated by a senior manager. The feedback proves useful to the team members in honing their skills, understanding where their erred in their role and the mistakes they made. The feedback proves useful in future assignment, promotions and incentives. Evaluation of the project manager is carried out in terms of; the project delivering the promised work, abiding by budgetary provisions, following established processes, and ability to manage men and resources.

Q. 6 What is RFP?

Ans. : Request for Proposal

Projects require resources, products and services that are not always available within the organization and for which the organization has to resort to external consultants, suppliers and agencies. To procure these products, resources and services the project manager has to correctly identify the needs of the project that cannot be met internally and then specify it in a document called the Request for Proposal or RFP.

A Request for Proposal (RFP) is a document that is used to solicit bids, quotes or proposals from prospective suppliers. On the other hand, a proposal is a document prepared by suppliers when there are different approaches for meeting the requirement of the organization. In the RFP the supplier may propose the use of various software, hardware and networking solutions to meet the requirements of the organization.

Developing an RFP is a time consuming process and begins by determining the project needs that cannot be fulfilled internally and for which the organization will need to avail the services of outside suppliers. The task of determining the needs that can be

Scanned by CamScanner

met internally and those that cannot be met need to be left with the project manager and the project team. Along with determining the needs for which external suppliers are required the project manager and his team should also determine when, where, and how much of these resources would be required. The idea behind an RFP is to prepare a level playing field for all the prospective suppliers by stating in clear terms the requirements of the organization and all other technical specifications. An unambiguous document enables all suppliers to send their proposals clearly and also enables the organization to compare and evaluate proposals of various suppliers. Although, price is an important factor in determining the choice of supplier there are other factors such as experience, technical capability, track record, financial stability, after sales service, etc which should be taken into account while determining the choice of supplier.

The organization should obtain proposals from a reasonable number of reputed suppliers. To invite proposals the organization should either directly contact reputed suppliers and send them the RFP or could release an advertisement in the newspaper, internet or the trade journal and get suppliers to send their proposals. The project manager should also invite prospective suppliers for a meeting and update them on the requirement and provide answer to their queries.

The elements of the supplier's proposal will include the following :

These items are a general framework and more or less should be a part of any proposal regardless of the type of work being done.

1. **Summary of project :** Identify the scope of work.
2. **Overall budget and costs :** Identify what the overall project budget looks like. At this point the IT professional will have to negotiate with the organization and give a written estimate of total cost.
3. **Breakdown of each task and associated costs :** Identify the specifics of each task and justify associated costs. Some costs may be fixed while others may be more variable. Some of the costs may be speculative and hence need to be indicated. Making sure the client knows this ahead of time will make it easier should unexpected costs arise.
4. **Payment Schedule :** States the manner in which the payment for the work undertaken is expected.
5. **Provisions :** Identify any provisions that are specific to the project. Provisions are an extremely important piece of the proposal as they allow the IT professional to set his terms in a legally-binding manner. The provisions included within the project proposal should include any that are specific to the project as well as any that are general to the business. Those provisions will be restated within the contract along with a general summary of the proposal.
6. **Signatures :** Even though the proposal likely won't be a legally binding contract once it has been negotiated and agreed upon, it is still good to sign off and have the client sign off on it as well.

Once, the proposals have been received the buying organization can begin the process of comparing, analyzing, and evaluating each proposal. The buying organization should prepare beforehand the criteria for selection of the supplier. As stated earlier, although price is an essential factor, other factors which could affect the outcome of the project should also be taken into consideration before selection of the supplier.

Q. 7 What is a Business Case? State the steps in developing the business case.

Ans. : Business case

The desire to utilize information technology to improve the efficiency and effectiveness of the organization has lead to an increase in the number of IT projects undertaken within organizations. However, the utilization of information technology without a thorough understanding of its risk and cost implications has lead to the failure of many IT projects.

The haphazard utilization of information technology merely because it is available is not judicious. The rampant failure of IT projects on account of the inability of projects to deliver returns as compared to the time and resources invested in them is an indicator of the need to develop some sort of analysis or feasibility study which would indicate the true value of the project for the organization. The business case arises out of this need and provides an analysis of the technical feasibility, costs, risks, returns, and organization value of various projects. It provides the basis on which informed decision on the projects can be taken.

Although, the business case may sound very similar to a project plan or budget it should not be mixed up with one. The business case is a document that provides the top management with all the information needed to select the projects that are to be funded.

Although creation of the business case document is very much similar to the feasibility report in most of the cases it is a separate document. Like the feasibility study the business case too helps the management in justifying the cost that will be incurred on the project and its return on investment.

The business case is built on the relevance of the business goals and objectives and the cost of the proposed technology that can get the organization there. The business case takes into account the cost of the solution, break-even point, return on investment, and maintenance cost. Along with the quantitative issues the business case may also address qualitative issues such as; working comfort, increased efficiency, improved morale, etc.

The business case is very similar to an investment proposal. Hence, the developer of the business case has to present compelling facts and figures in favour of the project and base his argument in the most logical manner. However, the business case

Scanned by CamScanner

developer in no way should digress from the facts of the project and the analysis should be as dispassionate as possible. A good business case should;

1. Detail all possible impacts, costs and benefits.
2. Be clear and logical in comparing the cost-benefit impact of each project alternative.
3. Include all pertinent information.
4. Systematically summarize all the findings.

Steps in developing the business case

Step I : Forming the team - developing the business case should not be a single man's responsibility. On the other hand, a team comprising of stakeholders, managers, users and IT specialists should be formed. The team so formed should bring in all the requisite knowledge, experience, information and expertise required to develop the business case. Of the team, the stakeholders are the people who are going to be affected by the project and hence their point of view needs to be presented in the business case document. Business managers bring in a higher level of perspective which is so essential for the business case. The users are the ones who understand the requirements that the project has to fulfil. While the IT specialists understand the risks and benefits associated with IT and should present their perspective in the business case.

Step II : Developing Measurable Organizational Value (MOV) - Measurable organizational value (MOV) is the IT projects overall goal and its measure of success. For any project the MOV should align with the organization's overall mission, objectives and goals. The term Measurable Organizational Value was coined by Jack Marchewka as an alternative to the more popular return on investment (ROI). According to Jack Marchewka the projects MOV should be:

1. Measurable
2. Provide value to the organization.
3. Agreed upon by all the core team members.
4. Verifiable at the end of the project.
5. Guide the project throughout its lifecycle.
6. Align with the organizations strategy and goals.

A clear MOV will enable the team to know where the project should go, it will be like the road that the whole project lifecycle should take. In case the project deviates from its path, the relevant decisions and adjustments will be based on the MOV which can be vital in achieving the overall goal.

Steps in developing the MOV

(i) **Identifying the desired area of impact :** The desired area of impact expected from the IT project is the primary reason for undertaking the project. To identify the desired area of impact the project manager should find out from the project sponsor how the idea of the project came about. Although, the answers could be vague it would provide the project manager with some elementary background as to how decisions are made in the organization.

(ii) **Identifying the desired value of the project :** The value that a project brings to an organization could be in terms of doing something faster, better, cheaper, or on increasing the market share. The project manager should identify and highlight what the project could potentially deliver to the organization.

(iii) **Developing an appropriate metric :** Once the desired value of the project has been identified it is time to quantify the value that would be delivered. So for example a proposed business information system would enable a company to process customer orders faster, the management would be better off in learning the reduced customer order cycle time. Therefore, the value should be quantified and expressed in terms of a metrics i.e. in terms of time, money, percentage or a specific value.

(iv) **Setting a time frame for achieving the MOV :** The time frame for achieving the MOV should also be mentioned. Some MOV's are immediately achievable on the completion of the project, for example the reduction in customer order cycle time, however, some like increase in market share may take time.

(v) **Verifying with stakeholders :** Getting the metric value and time frame verified and approved from the stakeholders adds value to claims made in the business case.

(vi) **Summarize the MOV in a clear and concise statement :** The MOV should be mentioned in a clear and concise statement.

Step III : Identifying Alternatives : All the alternative solutions to the problem or opportunity need to be delved upon in the business case. These alternatives should also enable the company to achieve the desired MOV. The alternative of maintaining status quo i.e. not doing anything and continuing with the present situation could also be a solution that needs to be looked into. However, the business case should put forth compelling reasons to bring about change and the cost that would be incurred in continuing with the existing process, systems or product. The cost incurred could be in terms of downtime, maintenance cost and systems failure.

Step IV : Defining Feasibility and Assess Risk : The feasibility and risk associated with each alternative solution should be analysed. Feasibility is the probability of successfully implementing an alternative while risk focuses on what can go wrong and what must go right.

Scanned by CamScanner

Feasibility and risk analysis will enable the project manager to identify alternatives that are not worth pursuing. Feasibility is viewed in terms of economic feasibility, technical feasibility, and organizational feasibility. Risk analysis focuses on its identification, assessment, and response.

Step V : **Defining Total Cost of Ownership :** The total cost of ownership of the application needs to be accounted for before any decision on implementing it is to be taken. Total cost of ownership is over and above the cost of purchasing or developing the application and as such includes cost of purchasing/developing, training, maintaining and supporting the application over its entire lifetime. Thus, total cost of ownership calculation is complex and hence the project manager has to authenticate his calculation with data sources, assumptions and methods for arriving at the cost.

Step VI : **Defining Total Benefits of Ownership :** The total benefits of ownership includes direct and indirect benefits associated with each alternative. Benefits could be in terms of increased efficiency, improved productivity, improved customer service, improved accuracy and efficiency, or improved decision making. However, not all benefits all easy to identify and quantify. Every alternative has certain tangible and intangible benefits. Tangible benefits are easy to identify and quantify as they lead to cost savings. On the other hand, though intangible benefits are identifiable they are difficult to quantify. The project manager should try and quantify intangible benefits by linking them to tangible benefits.

Step VII : **Analysing alternatives :** Once costs and benefits have been identified it is time to compare all the alternatives and arrive at one that best meets the requirement of the organization. Financial models such as ROI, Payback and Net Present Value and Scoring Models are used to analyse the alternatives.

Step VIII : **Recommend Solution :** After analysing each alternative the one that best suits the company should be recommended for approval.

Chapter 6 : Project Scheduling and Risk Management

Q. 1 State the benefits of WBS.

Ans. : Benefits of WBS

Dividing complex projects to simpler and manageable tasks is the main purpose of developing WBS. This method is used by project managers to simplify project execution as smaller work packages and activities are easier to manage. Following are the benefits for developing WBS in a project :

1. WBS serves as an input to key project management activities, namely, cost budgeting, resource planning, risk management, activity definition, and schedule planning.
2. WBS illustrates the project scope, so that every team member as well as stakeholder can have a better understanding of it.
3. WBS provides an accurate and readable project organization.
4. WBS facilitates accurate assignment of responsibilities to the project team.
5. WBS indicates the project milestones and control points.

Q. 2 Briefly describe the process of developing the project schedule.

Ans. : Process of developing the project schedule

Top level of the WBS offers high level deliverables each of which are decomposed into detailed and well defined deliverables known as work packages. These work packages are further decomposed to form activities in the project schedule. Activities or tasks are a key input of the project schedule. A list of all the activities to be included in the project schedule is tabulated. The activity list to be included on the project schedule includes the activity name, the activity code/identifier/number, and a brief description of the activity.

On the other hand, the activity attributes contains more schedule related information about each activity such as predecessor and successor activity, logical relationship between activities, resource requirement, constraints, dates, and assumptions related to each activity. The data for the preparation of the activity list and activity attribute is the WBS and WBS dictionary. As information on the requirements of various activities emerges it is updated in the activity attributes. Activity information is a necessary input for other processes and in activity sequencing, resource allocation, estimating duration, and developing the schedule. The project team should get the activity list and attributes reviewed by the project stakeholders before undertaking project scheduling.

A milestone is a significant event in the life of a project. A milestone in a project is accomplished after the completion of various activities. Although, the milestone has no duration, it marks the completion of various activities and is considered a key event. The primary purpose of setting milestones is that they prove useful in setting schedule goals and monitoring the progress of the project activities. Milestones might include completion of certain software modules, installation of hardware, etc.

Thus, milestone marks the completion of an important stage in the life of the project and the culmination of activities undertaken to reach that milestone. However, this does not mean

Scanned by CamScanner

that every project deliverable is a milestone. A milestone is a special event, something which is important for the project and takes it closer to its final destination. Project management is a balancing act - balancing scope, time and cost, necessarily in the same order. As we have seen the project team and the stakeholders first define the project's scope, then the schedule or time and finally the cost of the project. The order in which the three primary aspects of a project are addressed reflects the order of the processes in project time management.

The activity definition further defines the project scope, the activity sequencing further defines the time while activity resource estimating and activity duration estimating further define time and cost of the project. These four project time management processes form the basis of the project schedule. The most important resource in IT projects is its people and the biggest component of IT project budget is the cost incurred on them. This cost is calculated on an hourly basis and hence the time spent on the project needs to be monitored and managed effectively. In the truest sense, the adage "time is money" is applicable to IT projects.

Time management is the process of recording and controlling time spent by staff on the project. As time is a scarce and a costly resource within projects, each team member should record time spent undertaking project activities on a timesheet form. This will enable the project manager to monitor the amount of time spent undertaking each activity within the project. A timesheet register is also completed, providing a summary of the time spent on the project in total so that the project plan can always be kept up to date.

Project time management includes the processes required to accomplish the timely completion of the project. Project time management includes the following :

1. **Activity Definition :** This includes identifying the activities in the schedule that need to be performed to accomplish project deliverables. The activities or tasks are found in the project's work breakdown structure (WBS) and have expected duration, cost and resource requirement. The output of this process is an activity list, activity attributes, milestone list and project management plan updates.
2. **Activity Sequencing :** This includes identifying and documenting the dependencies amongst the schedule activities. The primary outputs of this process are the project schedule network diagram and the project documents updates.
3. **Activity Resource Estimating :** This includes estimating the type and quantities of each resource (people, software and hardware) required to perform each schedule activity. The primary outputs of this process are resource breakdown structure, activity resource requirements, and project document updates.
4. **Activity Duration Estimating :** This includes estimating the duration required by schedule activity to be completed. The primary outputs of this process are the activity duration plan and project document updates.
5. **Schedule Development :** This includes analyzing the activity sequences, durations, resources, and schedule constraints to create the project schedule. The output of this process is the project schedule.
6. **Schedule Control :** Controlling changes to the project schedule.

Although, these processes are shown to be discrete components with well defined boundaries and interfaces in reality they overlap and interact not only with each other but also with processes from other knowledge areas. Each process within the project time management needs the active participation of one or more than one person depending on the size and complexity of the project. The processes will need to be repeated for each phase if the project is divided into phases, if not it will need to be undertaken just once.

The six processes of project time management are preceded by a planning effort on the part of the project team which produces the schedule management plan.

The schedule management plan is responsible for developing and controlling the project schedule. The schedule management plan is a subsidiary of project management plan and includes the project time management processes and its various tools and techniques.

Simply put, schedule management plan involves determining the policies, procedures and documents that will be used for planning, executing and controlling the project schedule.

Q. 3 What purpose does the project schedule serve?

Ans. : Purpose of project schedule

Effective Planning and analyzing the plans from different perspectives increase the chances of completing the project on time and within budget. However, before setting the plan in motion it is necessary to work out a proper project schedule and plan the procurement activities listed in the plan. Just as every project must have a definitive set of deliverables which mark the closure of the project; it should have a designated start and finish date.

A project with no starting and finishing date is less likely to succeed. As far as possible the deadlines set for the project should be inflexible. Consider the Y2K bug problem surely this project implemented by every company had an inflexible deadline. Although, not all projects have such a demanding and inflexible deadline, however the project manager should set a realistic deadline to ensure that the project meets its logical end. A project missing its deadlines is a sure shot indication of poor planning and a lack of research.

Scanned by CamScanner

Perhaps one of the biggest challenge and area of conflict for any project manager is to deliver projects on time. Problems of schedule are common, easy to measure and remember. Project stakeholders and sponsors remember the project's scheduled completion date and quickly estimate the delay in the project without giving a thought to the approved changes in the project that may have lead to the overrun in date. Thus, time is one variable that has very less flexibility and needs to be respected no matter what happens on the project. Hence, with all possibilities of schedule conflicts it is important for project managers to adopt good project time management.

Project Schedule serves the following purpose :

1. Project schedule simplifies the project plan such that it can be understood by all those concerned and initiates prompt action.
2. Project schedule validates the time objective by verifying the deadlines set for tasks.
3. The implications of a delay in a particular task can be gauged and proper corrective action can be initiated to avoid repercussions.
4. Project schedule enables the project manager to identify and focus on those activities that could prove to be bottleneck in the future.
5. Project schedule serves as a basis for project monitoring and control.
6. Project schedule ensures the optimum utilization of resources and helps in resource levelling by working on the floats.
7. Project schedule identifies scheduling constraints and enables the project manager to work within the framework of the constraints.

Q. 4 What is Gantt chart? How is it created?

Ans. : Gantt chart

A Gantt chart or more popularly known as the bar chart is one of the most commonly used project management tools and are used in tracking project schedules. Although, the Gantt charts were originally devised by Karol Adamiecki in the 1890's, they were later on modified by Henri Gantt, an American engineer and management consultant, and hence they are widely known in western countries as Gantt charts.

Gantt chart provide a standard format for displaying project schedule information by listing project activities and their corresponding start and finish dates in a calendar format. These charts are also referred to as bar charts as activity start and finish dates are shown by horizontal bars. The primary reason for the popularity of the Gantt charts is because they are easy to plot and simple to understand. The Gantt adopts a very simple process of showing activities (tasks or events) displayed against time.

A Gantt chart is a graphical representation of a project plan on a time scale. The X axis of the chart represents time i.e. project duration while the Y axis represents activities in the form of rectangular bars. The position and length of the bar reflects the start date, duration of the activity and the end date of the activity. Each horizontal bar represents one specific activity to be performed and the beginning and end of each bar represents the commencement and completion of the activity.

The Gantt chart enables the user to :

1. Identify the various activities in the project
2. Identify the beginning and end of each activity
3. Identify the duration of each activity
4. Identify the overlapping of activities
5. Identify the start and end date of activities

Therefore, in simple terms a Gantt chart enables the user to understand what has to be done i.e. the activities to be performed and when it is to be done i.e. the schedule of the activities.

Creating Gantt chart

Gantt charts are widely used in business to describe and monitor all kinds of projects according to the rules of project management. The basic requirement of a Gantt chart is a detailed project plan. A project plan is a series of interdependent tasks that need to be performed in a particular order, for example you create a WBS before developing a project scope statement. Project plans have a specific start date, corresponding to the start of the first task, and a specific end date, corresponding to the end of the last task. The project plan can be created by using the Work Breakdown Structure, which splits tasks into sub tasks and creates a task hierarchy.

Once the project plan is complete the project manager can create the Gantt chart. The X axis will have specific units of time designated such as days, weeks, etc. Activities on Y Axis are listed in order of the earliest start on the project. Activities are first listed out vertically in a logical sequence of execution. Tentative duration of each activity is estimated based on the availability of resources. Activities on the Gantt chart should coincide with the activities on the WBS which in turn should coincide with the activity list and milestone list.

The scheduling starts from the project start date and tasks start as soon as possible which means that the project finishes at the earliest possible date. However, while scheduling the project manager has to keep in mind the number of working days in the week and the number of working hours in the day.

The tasks are named and duration and resources are assigned to each task. The project manager will need to specify the dependencies between the project tasks. Any change in the timing of a task affects all the tasks that depend on it. If a task runs ahead of schedule, the Gantt should recalculate the dates of all the tasks that depend on it in order to take advantage of the time gained.

Conversely, if a task is delayed, all the tasks that depend on it are rescheduled, which may or may not impact the overall end date

Scanned by CamScanner

of the project. The Gantt chart for IT project management contains milestones, summary tasks, individual task duration and arrows showing task dependencies.

Example

Activity No	Activity Description	Duration in Days	Remarks
1	Task1	5 days	
2	Task2	2 days	
3	Task3	6 days	Starts with activity 2
4	Task4	4 days	Starts 2 days after activity 3 or after completion of activity 2
5	Task5	4 days	Starts after completion of activity 4
6	Task6	12 days	Starts after completion of activity 5
7	Task7	3 days	Starts after completion of activity 6
8	Task8	2 days	Starts after completion of activity 5
9	Task9	5 days	Starts after completion of activity 6
10	Task10	12 days	Starts 3 days after starting of activity 9
11	Task11	4 days	Starts with activity 10
12	Task12	12 days	Starts 3 days after starting of activity 11
13	Task13	6 days	Starts 1 days after starting of activity 12
14	Task14	5 days	Starts after completion of activity 13
15	Task15	4 days	Starts after completion of activity 14

Total duration without considering overlapping of activities = 86 days.

Actual duration after considering the overlapping of activities = 86 – 37 = 49 days.

Q. 5 Write short note on : Milestone Chart

Ans. : Milestone chart

Milestones, are important part of the project schedule especially for large projects. Project managers like to focus on meeting milestones so as to emphasize the project accomplishment. Milestone charts are similar to Gantt Charts only that they highlight the scheduled completion of major deliverables and key external interfaces.

The Gantt chart can also be made to incorporate the milestones within the project. Milestones are marked by the completion of various project activities/tasks and hence are created by entering them as tasks with zero duration.

Q. 6 What is a critical path? Why is the longest path the critical path?

Ans. : Critical path

The critical path denotes the sequence of events that determine the project completion date. It is the longest path from project start to completion. Every project has multiple paths from project start to completion but there is only one path that takes longer than any other path in the project network diagram and this is our critical path.

A path is series of continuous activities thought the network that start from the initial node of the network to its final node. Now, every network has multiple paths that start from the initial node and culminate at the final node. However, all these paths are not critical paths. The critical path is the longest path.

Why is the longest path the critical path?

This is because a delay in any activity on this path is going to lead to a delay in the completion of the project. However, a project could have more than one paths vying for the title of critical path. The Critical path is the continuous chain of critical activities in a network diagram. It is the longest path starting from start to completion and is shown by a thick line or double lines crossing the activity line in the network diagram.

In some cases a project may have more than one critical path. This means that there may be two or more than two paths that may be consuming the same time for their completion. In such case all these paths are considered to be critical to the completion of the project and are thus critical paths of the project.

Q. 7 What is a float? State the types of float.

Ans. : Float

The activities not on the critical path can be delayed without any impact on the project or succeeding activity, although there is a

Scanned by CamScanner

limit to the amount of time they can be delayed. This limit is the called the float.

There are four types of floats:

I. **Total Float :** The total float of an activity represents the time by which the activity can be delayed without delaying the completion of the project. Thus, it refers to the amount of free time available with the activity that can be used for altering the early start of the activity. Total float is the difference between the total time available for the performance of the activity and the time required for its performance.

II. **Interfering Float :** There are chances that the utilization of the total float of an activity may affect the float times of other activities in the network. Thus, that part of the total float that causes a reduction in the float of the successor activity is called as interfering float.

III. **Free Float :** As the name indicates free float is that part of the total float that can be used without affecting the float of the succeeding activities. Thus, it is that quantum of the total float which can be consumed without affecting the early start of the succeeding activity.

IV. **Independent Float** : The independent float time of an activity is that amount of float time that can be used without affecting the preceding or the succeeding activity. Thus, independent float represents the amount of time available with an activity when its preceding activity are completed at their latest finish time and the succeeding activities begin at their earliest time.

This invariably means that the activity under consideration is left with minimum time for its completion. Any time available with the activity over the minimum time allotted for its completion is called as independent float.

Q. 8 Explain the three time estimates required for each activity under PERT.

Ans. :

The three most commonly used network techniques are Critical Path Method (CPM), Project Evaluation and Review Technique (PERT) and Precedence Diagramming Method (PDM) also called as Precedence Network Analysis (PNA). These network techniques generate time oriented diagram having activities organized into a logical order. The common features of all three techniques are that :

(i) They make use of the network model for depicting the time plan of the project.

(ii) Apply critical path concept for determining project duration and identifying critical activities

(iii) Employ network analysis techniques for controlling project time objectives.

However, each of these techniques has a distinct model of application.

CPM (Critical Path Method) and PERT (Programme Evaluation and Review Technique) are network techniques used for planning, scheduling and executing large projects which require coordination and execution of a variety of complex and large number of activities. These activities have to be completed within a specified time, cost and meeting the performance standards. CPM and PERT share in common the determination of sequence of activities, also called critical path and are based on the network representation of activities and their scheduling that determines the most critical activities to be controlled so as to meet the completion date of the project.

(i) CPM is best suited for developing the sub-networks of sub-groups and tasks having activities with deterministic single time duration. The sub-networks can then be systematically integrated using PNA technique.

(ii) PERT is useful for project involving uncertainties. In such cases, probabilistic approach of three times is used for assessing the activity duration.

(iii) PDM/PNA is the most commonly used method for time planning.

(iv) GERT is rarely used in project management.

CPM

1. This technique was developed in connection with a construction and maintenance project in which duration of each activity was known with certainty.
2. It is suitable for establishing a trade-off for optimum balancing between schedule time and cost of the project.
3. It is used for completion of projects involving activities of repetitive nature.

PERT

1. It assumes a probability distribution for the duration of each activity. Thus completion time estimates for all of the activities are needed.
2. To perform PERT analysis on a project, the emphasis is given on the completion of a task rather than the activities required to be performed to reach to a particular event or task. Thus, it is also called an event –oriented technique.
3. It is used for one time projects involving activities of non-repetitive nature (i.e. activities which may never have been performed before) in which time estimates are uncertain, such as redesigning an assembly line or installing a new information system.
4. It helps in identifying critical areas in a project so that necessary adjustment can be made to meet the scheduled completion date of the project.

Scanned by CamScanner

Q. 9 Differentiate between CPM and PERT.

Ans. : Difference between CPM and PERT.

There are three ways in which CPM type of network differs from the PERT type.

(i) CPM is built from activities instead of events

(ii) Time is related to costs

(iii) No allowance is made for uncertainty in the estimates of their durations

Factors	CPM	PERT
Application	Deterministic Projects	Projects involving uncertainties.
Model emphasis	Activity Oriented	Event Oriented
Activity duration estimation	Single time	Three times
Time-cost trade-off	Feasible	Not Feasible
Resources optimization	Feasible	Not Feasible
Technique Complexity	Simple	Difficult

Q. 10 What is software risk?

Ans. : Software risk

Risks are defined as the various uncertain events that happen due to change in the various factors in the software development. Risks are conditions, which may or may not arise, that can have a positive or negative effect on the project's ability to reach its set objective. Although, every project has some degree of risk, the risk quotient identified with IT projects is even higher given the speed with which IT technology changes. New technology may come along and outdate the technology used in the project even before the project has been implemented.

However, most risks can be identified and planned for. Thus most risks are foreseeable events which make it possible for the project team to make adequate arrangements to neutralize the negative effect they may have on the project. The risk management plan is an integral component of the overall project management plan.

During the development of the risk, the major event is the identification of risk and its management. The risk analysis process starts with detailing the risks under the various risk analysis factors as mitigate, monitor, and manage.

The best strategy to overcome risk is to anticipate risks in the project at the earliest so that the effect of the risks on the project is minimized by the risk management plan. To identify risks it is necessary to categorise them. There are primarily three categories of risk :

I. **Project Risks :** Project risks concerns various forms schedule, resources, budgetary and user requirement related issues.

II. **Technical Risks :** Technical risks concern various forms of design risk, implementation risk, interfacing, performance, and maintenance problems.

III. **Business Risks :** Business risks are various, project being redundant on completion, exceeding budgetary provisions, etc.

Q. 11 Write a note on risk identification.

Ans. : Risk identification

Risk is an integral part of an IT projects and therefore risk identification at the earliest becomes all the more critical. Risk identification is an iterative process, wherein the project manager and his team are always on the lookout for risk that may sneak into the project or may have already sneaked in.

Risk identification needs to be done throughout the project lifecycle, because as the project develops new risk is likely to emanate and pose a threat to the project. An idealistic situation would be one in which all the likely risks could be identified before the execution of the project but the reality is that new risk emanate as the project progresses and hence the project manager and his team have to always be alert and identify risk and nub them in their bud.

i. To identify the risks are always the biggest and the toughest task in project management.

ii. The risk can be categorized on the basis of the size of the project, and the software development, the risk identified is then modified.

iii. The risks which create a business impact due to the constraints imposed by management or the marketplace during the development phase till the deployment phase.

iv. As the customer played the critical role in the project development being the end user, the various characteristics related to the customer for communication are very important.

v. The exact requirement gathering and the analysis to be done for the development of the project at the Software Development Life Cycle has to be determined to avoid the risks.

vi. Good quality tools should be used and made available for the developers.

vii. New advances in the area of technology and the project development has to be studied for a perfect time lined software development.

viii. The major risks are associated if the staff and the team are not as per the required project.

Scanned by CamScanner

Risk Components and Drivers

The risk components are defined and identified on the criteria of the performance, support, cost and the schedule of the project :

1. **Performance risk :** If the specifications don't meet the final requirements of the project.
2. **Support risk :** The support the application developer provides after deployment of the project
3. **Cost risk :** To maintain the time and the cost of the project.
4. **Schedule risk :** The schedule and the parameters of deliverable should be maintained

There are several tools available that can help the project manager in identifying risks.

1. **Project Documents :** Project documents is the first place to start looking for project risks. The project scope statement, WBS, project plan, resource requirement, etc will help identify many risks associated with the project.
2. **SWOT Analysis :** Similar to an organizational SWOT, the project is examined in terms of its strengths, weaknesses, opportunities and threats and areas are identified where the project could fail and where the project could improve. SWOT is a great technique when it comes to resources and requirements not used before.
3. **Brainstorming :** A brainstorming session is one in which the project team sits together and thinks up with as many risks as possible. A brainstorming session is one in which the participants can let their imagination go wild, there is no restriction on the number of risks to be identified on the other hand the team is encouraged to anything that remotely resembles a risk.
4. **Delphi Technique :** Although brainstorming sessions do manage to identify the likely risks, a few participants may shy from identifying risks on account of personal reasons. Therefore, to overcome these limitations the Delphi Technique is adopted. The Delphi technique comprises of anonymous surveys conducted amongst team members to build consensus on risk events.
5. **Assumption analysis :** Every project is based on certain assumptions. However, these assumptions need to be tested to ensure their validity. Assumptions not tested have the potential to become risk in the future. Therefore, these assumptions need to be tested, researched and the results confirmed as to how they would be affecting the project. An assumption log should form a part of the project documents.
6. **Root cause analysis :** This analysis examines an effect that the project is experiencing and then goes to the root of the effect to identify its cause.

All the identified risks need to be entered into a risk register. The risk register will contain a log of all the risks and its impact on the project.

Q. 12 Write a note on risk planning and projection.

Ans. :

Risk planning and projection

Once the risk has been identified and analyzed the project manager can decide on the manner in which the risk can be dealt with. The response to risk depends on the type of risk. However, all risk should be documented along with the chosen response and its outcome in the risk register for future reference.

The response to risk depends on its type i.e. negative risk or positive risk. The three most common responses to negative risk are:

1. **Avoidance :** The first natural response to any risk to find ways to avoid it. This would mean creating workarounds, adjusting project objectives, or changing the project schedule, in short whatever required to avoid the risk.
2. **Transference :** The risk is transferred to a third party that is in a better position to manage the risk. Thus, instead of trying to manage the risk in-house the risk is transferred to a third party.
3. **Mitigation :** Here the project manager is left with no alternative but to tackle the risk in-house. This would mean spending extra time and money to reduce the probability and impact of the risk.

The three most common responses to positive risks are:

1. **Exploitation :** The project manager tries to gain maximum benefits from the positive risk. The idea here to utilize the risk to maximum.
2. **Enhancement :** The project manager creates an environment that is ideal for the positive risk to happen.
3. **Sharing :** The positive risk is shared by the project team with another entity to gain maximum benefits from the risk. The project team may not be capable of leveraging maximum benefits from the risk and hence feels that by bringing in a third party it may be able to derive maximum.

In addition to the above given responses there is another response that is suitable for both positive as well negative risk and this is the response of Acceptance. Positive as well as negative risks with very low impact should be accepted as it is.

The steps to determine the overall consequences of a risk :

1. The project component which can cause the risk needs to be identified.

Scanned by CamScanner

2. Using Fig. 6.1, the impact for each component has to be identified.
3. The overall risk exposure, RE, is determined using the following relationship RE = P x C
4. Where P is the probability of occurrence for a risk, and C is the cost to the project should the risk occur.

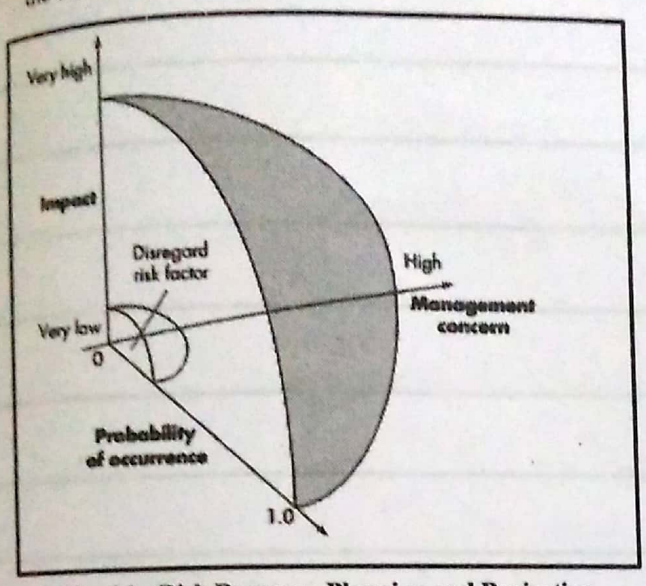


Fig. 6.1 : Risk Response Planning and Projection

For example, assume that the software team defines a project risk in the following manner:

1. Risk Identification. Only 60 integrated into the application. The remaining functionality will have to be custom developed.
2. Risk Probability. 70% (likely).
3. Risk Impact. 65 reusable software components were planned. If only 60 percent can be used, 22 components would have to be developed from scratch (in addition to other custom software that has been scheduled for development). Since the average component is100 LOC and local data indicate that the software engineering cost for each LOC is Rs.160,the overall cost (impact) to develop the components would be 22 x 100 x 160 = Rs. 3,52,000.

   Risk Exposure. RE = 0.70 x Rs. 352000 ~ Rs. 246400.

Q. 13 Write a note on RMMM.

Ans. :

RMMM

An effective strategy for dealing with risk must consider three issues

1. Risk mitigation (i.e., avoidance)
2. Risk monitoring
3. Risk management and contingency planning

Need for RMMM

1. Mitigate all the requirements and the causes that are under our control before the start of the project.
2. After the start of the project, the deliverables have to be done on the deadlines by the PERT.
3. The development of various groups has to be done by proper and timely communications between the teams.
4. The various documentations have to be prepared at the time of the deployment of the project.
5. The peer review, formal reviews and walkthroughs to be conducted for the project.
6. During risk monitoring, the project manager monitors factors that may provide an indication of whether a risk is becoming more or less likely to identify its intensity.
7. Risk management and its removal plan to be prepared and has to come to the reality of development.
8. RMMM steps add to an extra project cost during development and deployment.
9. Large projects may have identified 40 – 50 risks
10. Risk is identified with the various factors related to software projects.
11. Risk can be developed after the deployment of the project also.

❑❑❑

Scanned by CamScanner