Biyani's Think Tank

Concept based notes

Software Engineering

(MCA)

Professor
Dr. Madhu Sharma
Biyani Institute of Science and Management, Jaipur

Published by :

Think Tanks
Biyani Group of Colleges

Sector-3, Vidhyadhar Nagar,
Jaipur-302 023 (Rajasthan)
Ph : 0141-2338371, 2338591-95 • Fax : 0141-2338007
E-mail : acad@biyanicolleges.org
Website :www.gurukpo.com; www.biyanicolleges.org

Edition: 2013

Laser Type Set by:
Biyani College Printing Department

Index

Introduction

Q.1 What do you mean by software?

Ans: A software system or software is:

(i) Instructions/Computer programs that when executed provide desired function & performance.

(ii) Data structures that enable the programs to adequately manipulate information.

(iii) Documents that describe the operation and use of the programs.

We can say that a software system consists of a number of separate programs, configuration files, which are used to setup these programs, system documentation, which describes the structure of the system & user documentation which explains how to use the system & website for users to download recent product information.

Q.2 What are the characteristics of Software?

Ans: For a better understanding about the software and hence software engineering, the characteristics of the software are required to be discussed. Since, software is a logical system (whereas, hardware is a physical system), its characteristics are different from hardware:

1. Software is developed or engineered; it is not manufactured in the classical sense.
2. Software doesn't "wear out".
3. Most of the software continues to be custom built or customizable.

Q.3 Compare software development and hardware manufacture?

Ans: Similarities between software development & hardware manufacture:

- High quality is achieved through good design
- Both activities depend on people and their requirements.

Dissimilarities between software development & hardware manufacture:

- The manufacturing phase of hardware can have quality problems which couldn't be rectified as easily as in software development phase.

- Though both activities depend on people, but the relationship between the people applied and work accomplished is entirely different. As for hardware manufacturing market survey and competitors policies are to be considered on priority, whereas software development concentrates more on the user requirements.
- Both activities target for a product as a result, but the approaches followed are entirely different. (As there are SDLC set of phases for software development, whereas the product manufacturing involves different phases like consumer tendency, market analysis etc.)
- Cost of manufacturing a hardware product could be estimated exactly & easily, whereas it's comparatively difficult to estimate exactly in case of software development.
- Software doesn't "Wear out". The relationship of failure rate with time for a hardware is like a "bathtub curve" As shown in Fig 1.1

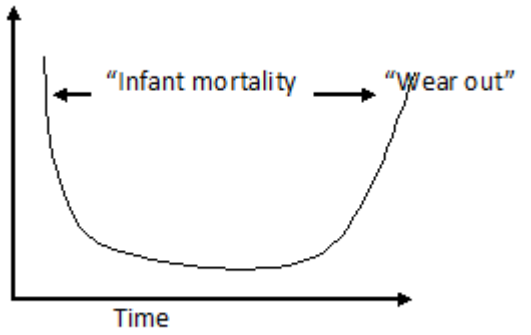


Fig 1.1 Failure curve for hardware

This curve shows that hardware show high rate of failure in it's early stage due to design or manufacturing defects, when there defects are corrected, the failure rate deeps i.e. very low & steady for a period of time. But as the time passes, due to environmental effects like temperature, dust, moisture, the hardware begins to wear out. In a broader way, as comparing software with hardware, we can say that the software doesn't wear out due to environmental conditions (as in case of hardware). Theoretically the failure rate v/s time curve for software is an ideal curve as shown in figure 1.2

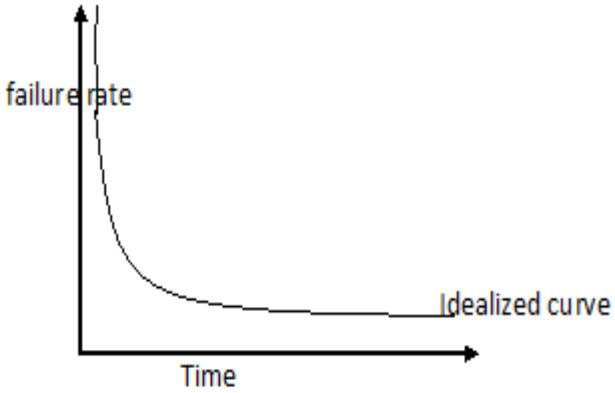


Figure 1.2 Idealized curve for Software

Here in the initial cycle of a software, undiscovered defects cause high failure rates, but if they are corrected (i.e. made errorfree), the curve becomes flat & ideal with very less failure rate. This shows that software doesn't wear out, but the fact is that, it deteriorates by the attack of malicious software (virus, Trojan horse etc.) and also by time the user's demands changes and updations are required which leads to further changes & modifications causing to increase in failure rates as shown in figure 1.3.

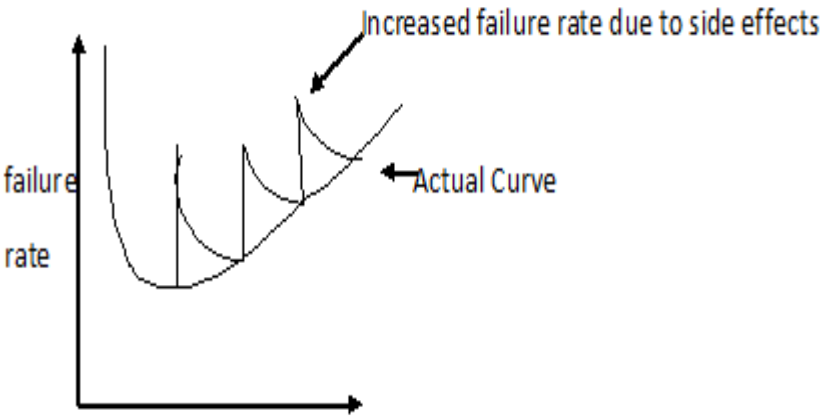


Figure 1.3 Actual curve for Software

- The software industry is moving towards component based assembly. Most of the software continues to be customer built. As in case of hardware, we have well defined set of IC or chips (Integrated) and screws and other components and that could be used to design different hardware (reusability of hardware components). In case of software, library routines & subroutines are to be developed for reusability. A software component should be designed & implemented so that is can be reused in many other different programs.

Q.4 What are Software Myths?

Ans: Most, knowledgeable professionals have recognized myths or beliefs (untrue beliefs or explanations) which are misguiding the attitudes (of the users) causing serious problems for managers & technical persons. Different types of Myths relevant to the software are as classified in figure 1.4.

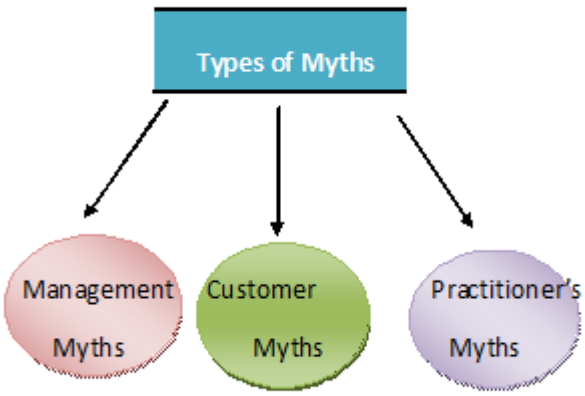


Figure 1.4 Software Myths

(i)Management Myths:
Myth 1:
We have all the standards & procedures available for building software i.e the software developer have everything what is reqd.
Reality:
- Software practitioners are not aware about the existence of all those standards.
- Those practices might be outdated or not according to the current /modern software engineering practices.
- Also all the pre-existing procedures, are not complete.

Myth 2:
The Addition of latest hardware systems would improve the software development.
Reality:
- The role of latest hardware is not too high for the quality software development; instead (CASE) Computer aided software engineering tools are more important than hardware for a good & quality productivity.
- Also the available hardware resources are not used effectively.

Myth 3:
Management thinks that, addition of more people and programmers in software development could help in meeting the deadlines of the project (If it is lagging behind)
Reality:
- Software development is not, a mechanistic process like manufacturing; here addition of people in late phases may reduce the amount of time to be used for the productive development, as the newcomers would take time of existing developers for the explanations & understanding for the project. But, addition of people done in a planned & well-coordinated manner could be helpful in the project completion.

(ii)Customer Myths:
Customer could be the direct users of the software, a technical group, marketing /sales department, or some other company. The customer possesses the myths which lead to the false expectations (by the customer) & hence create dissatisfaction with the developer.
Myth 1:
A general statement of objectives is sufficient to begin writing programs (software development) and the details of the objectives could be done later.
Reality:
A formal & detailed description of the information domain function, behaviors performance, interfaces, design constraints and validation criteria is essential. A thorough communication between customer & developer is must.
Myth 2:
Project requirement continually change, but, change, can be easily accommodated due to the flexible nature of the software.
Reality:
Changes are made till the last phases of the software development but the cost of making those changes increase with the later stages of development. A detailed analysis of user requirements should be done to minimize the change requirement. Fig 1.5 shows the cost of change with respect to the phases of development.

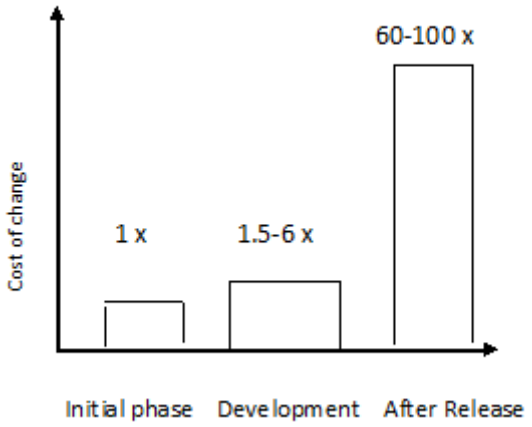


Figure: 1.5 Cost of change

(iii)Practitioner's Myths:

Myths 1:

They believe that their job is completed with the program writing & get it worked.

Reality:

It's a reality that 60-80% of all the efforts go in the maintenance phase (as after the release) of the software. The efforts are required, when the product is delivered first time to the customer.

Myths 2:

There is no (other) way of accessing the quality of the program, after it is being made to run.

Reality:

The formal technical review of the project is an effective software quality assurance mechanism. These reviews are the quality filters and more approachable than the testing.

Myth 3:

The working program is the only deliverable work product for a successful project.

Reality:

The working program is not only sufficient, instead proper documents manuals and handouts are also reqd. to be supplied for the guidance & software support.

Myth4:

Software Engineering will make us create voluminous & unnecessary document & invariably slow us down.

Reality: Software Engineering is not about creating documents, instead it is there for creating quality and better quality leads to the reduced rework & this reads to a faster delivery of the product.

Q.5 What are various Software Applications?

Ans: Software may be applied in any situation for which a specified set of procedural steps (i.e. an algorithm) has been defined. Information content & determinacy are important factors in determining the nature of a software application.
Following are the categories of the software with their corresponding applications.

1. System Software: It is a collection of programs written to save other programs eg. Compliers, Editor, File Management utilities, Operating System. These are characterized by heavy interaction with computer hardware and heavy usage by multiple users.

2. Real Time Software: Software that monitors and analyzes the real world problems and meets strictly the timelines in execution.

3. Business Software: System like payroll accounts receivable/payable, inventory, MIS software for providing information are included in business software.

4. Engineering & Scientific software: Applications used specifically for engineering, research, simulation automated manufacturing etc. are included in such software.

5. Embedded Software: Artificial Intelligence based (home applications) intelligent systems like keypad or microwave oven, automated washing machines etc.

6. Personal computer Software: Used mainly for entertainment (like media players), word processing, spreadsheets , graphics etc.

7. Web based software: These mainly incorporate the instructions based on CGI, HTML, PERL etc.

8. Artificial Intelligence Software: To solve complex specific problems eg: expert system, knowledge-based systems, pattern recognition etc.

Q.6 Define Software Engineering.

Ans: Software Engineering (SE) is the establishment & use of sound engineering principles in order to obtain economic, reliable and efficient software. Software Engineering is the application of a systematic, disciplined, quantifiable approach to the development, operation & maintenance of software. Software Engineering is layered technology as shown in Figure 1.6.

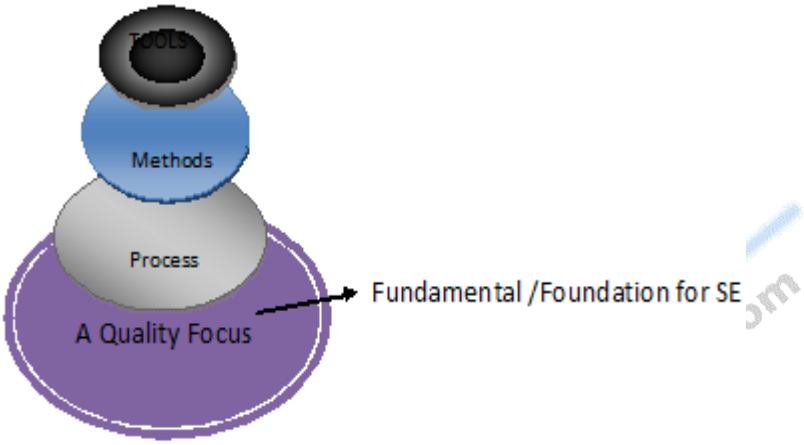


Figure 1.4 SE layers

Software Process Models

Q.1 Explain software process.

Ans: The software process can be shown as in Fig 2.1:

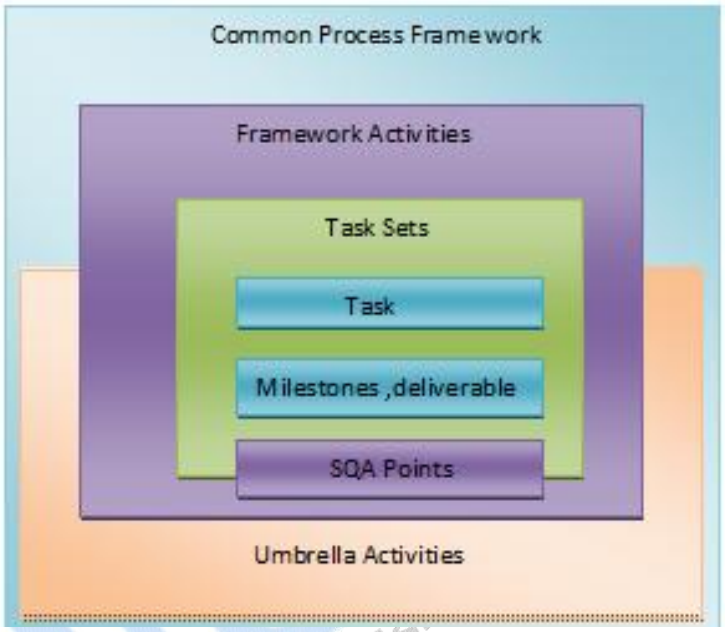


Figure: 2.1 Software Process

A Common Process Framework is established by defining a small number of framework activities that are applicable to all software projects, regardless of their size or complexity. Task sets are the collection of software engineering work tasks, project mile stones, work products & quality assurance points. Umbrella activities majorly includes software quality assurance, software configuration mgmt. & measurement, these activities are independent of any other framework activity & occur throughout the process.

Q.2 What is the basic structure of software development process?

Ans: A software development process model is an abstract representation of a software Process or is a brief overview of the software process. Basic structure of the software development process is shown in Fig 2.2.

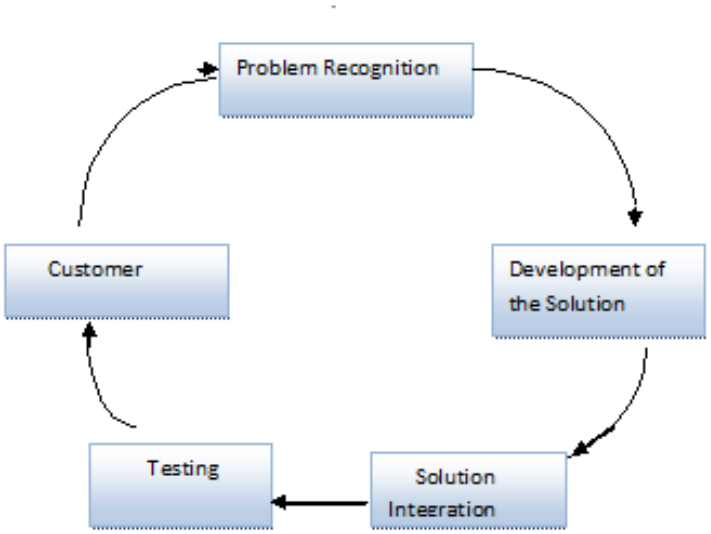


Figure: 2.2 Basic Structure of the Software Development Process

The process development is similar to SDLC.

Q.3 What are the different categories of process models?

Ans: There are many process models which are used in different areas. The waterfall model & the evolutionary models are commonly used for the practical development. The other models are used in the specific fields. The model categorization is shown in Fig. 2.3

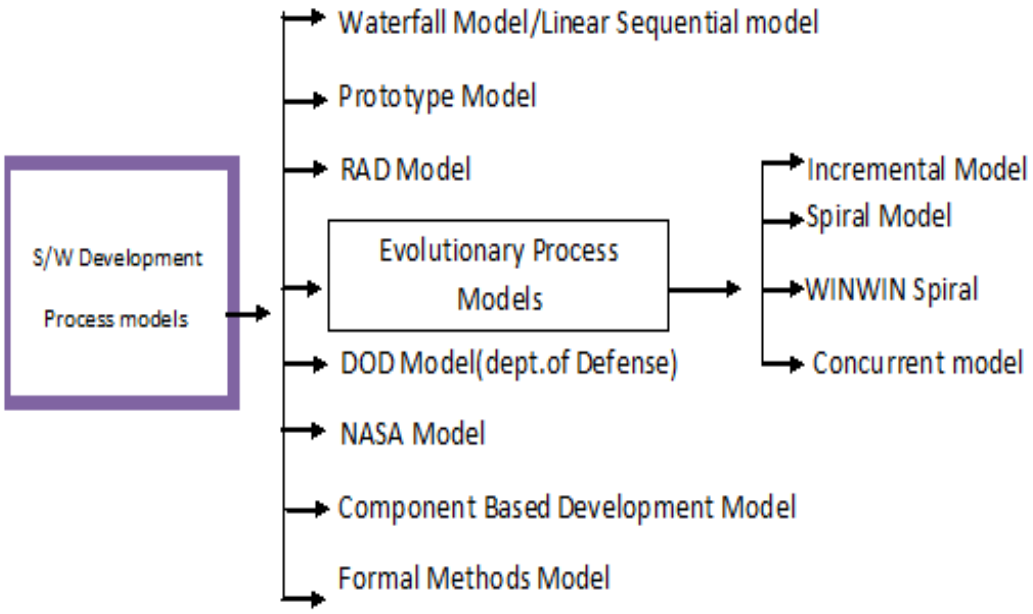


Figure: 2.3 Process Models Categories

Q.4 Explain Waterfall model / linear Sequential Model.

Ans: Waterfall model is:

- The simplest oldest process model
- Best suited where requirements can be clearly defined.
- Steps of development are followed consecutively
- Each step involves detailed activities
- Each step produces an output which becomes the input of the next step.

Major phases/steps of the waterfall model are shown in fig 2.4

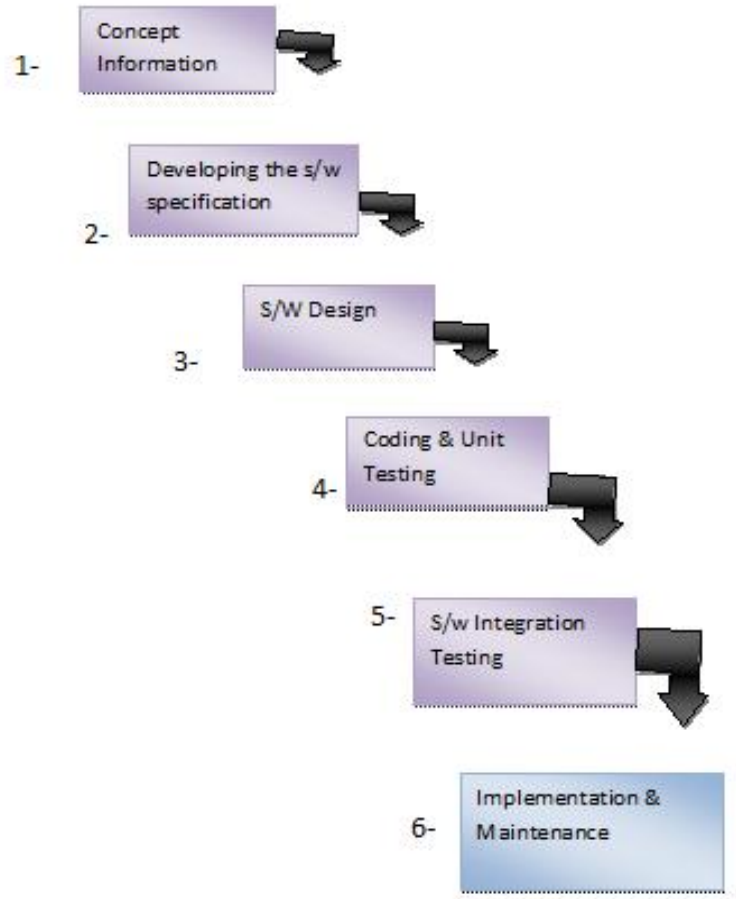


Figure: 2.4 Waterfall Model

Phase1-Concept formation

This phase includes-

(a) Problem definition: Tells the requirements of the user. Project goals, scope & resource limits are identified.

(b) Feasibility Study: Provides one or more conceptual solution. Solutions must be proved feasible & a preferred sol must be accepted.

Outputs:

- User requirements
- Proposed solution
- Project goals & scope
- Terms, condition & restrictions
- Tentative dates of project start & completion

Phase: 2 Developing the software specification

A Detail analysis model with high-level description of requirements is produced

Outputs:

- Requirements Analysis Model
- Revise to project goals & cost Benefit Estimates
- A SRS (Software Requirement Specification)
- A project development plan
- A Test plan

Phase: 3 Software Design

Two steps are there:-

(a) Preliminary : Main Architecture is proposed

(b) Detailed: Detailed database & program modules and procedures are designed & documented.

Outputs:

- Detailed User Procedures
- User Manual
- SDS (Software Design Specification)

Phase: 4 Coding & Unit Testing

- Programs are written in specific programming language (as mentioned in SRS)
- Individual components are tested

Outputs:

- A Test Document
- Tested Modules

Phase: 5 Software Integration & Testing

Individual units are integrated & then tested. The complete testing procedure is shown in Fig 2.5.

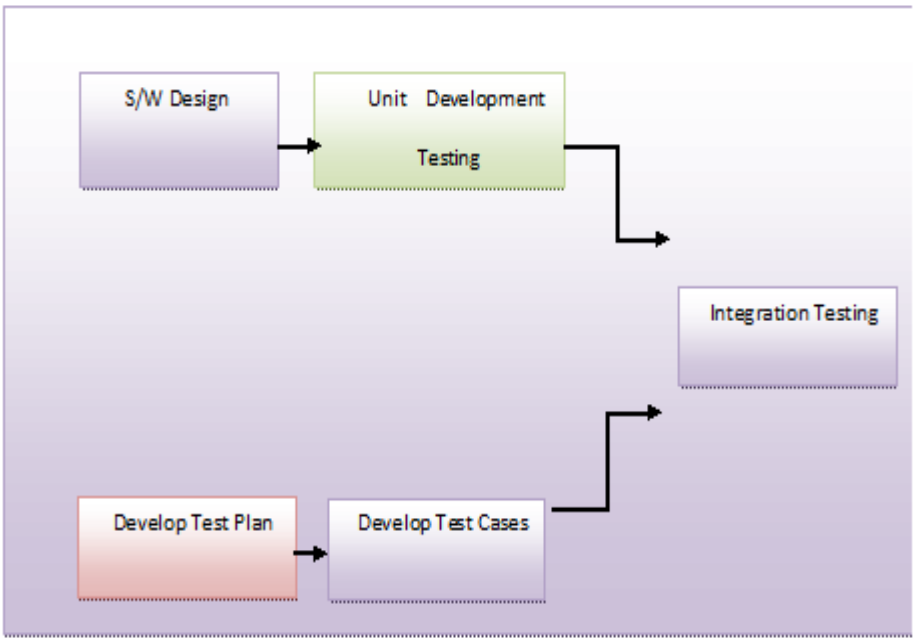


Figure: 2.5 Testing Procedure

Output:
A complete integrated & tested software for the user.

Phase: 6 Implementation & Maintenance :
Software is installed at user location & if any deficiencies or errors are found, they are removed. If the changes are not minor, then requirements of a new system arise.

Model Summary :

- It is a step –by-step approach
- Each step should be well-defined
- Each step creates a definite product
- Product generated from each phase becomes the basics for the next step.
- Verification & validation of each step can be done followed by certification
- Clear demonstration between end & beginning of a phase.
- The outputs is work product obtained as: SRS(Software Requirement specification) , SDS (Software Design Specification) , project plan , test plan, and code.

Advantages

- Model has well-defined steps with well-defined outputs.
- It recognizes the sequence of software engineering activities.

Disadvantages

- Because of greater wait, this model not in much use.
- Not easy to find all requirements at the beginning stage.
- Greater customer patience is required.
- Less interaction of software developers & users
- Inflexible in nature, as phases are assumed as compartmentalized activities.

Q.5 Explain Prototype Model/Rapid Prototype Model.

Ans: Prototype Model is also called as Rapid Prototype model.

- It removes the limitation of the waterfall model.
- A Basic model or prototype is developed before the actual model.
- It is built very rapidly & fast, so that it can be checked time to time by the customer.

Fig 2.6 shows the phases of prototype model.

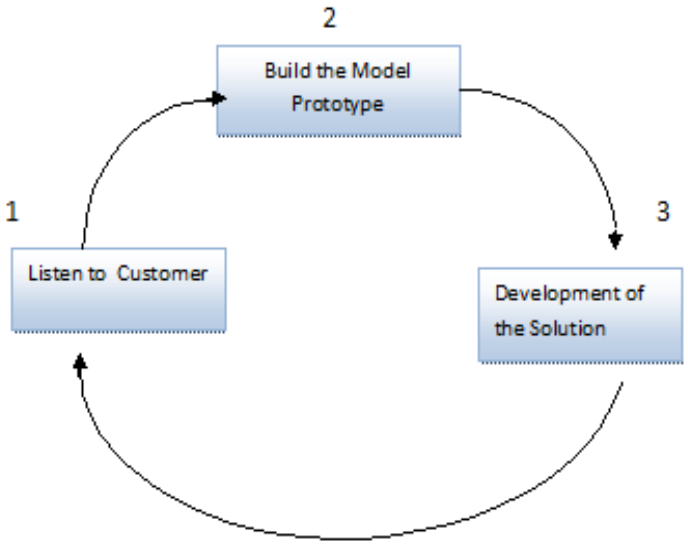


Figure: 2.6 Prototype Model

Phase 1: Listen to Customers

- All requirements of the user are discussed & identified
- Goals are set

Output:

- An agreement document
- List of all requirements

Phase 2: Build the Model

- Model is quickly designed according to user's requirements

Output:

- A Prototype for the customer is ready.

Phase 3: Customer Evaluation

- System is evaluated & tested by customer

Output:

- Feedback from customer to developer
- User satisfaction or acceptance

Prototyping is a practice of building a version of software quickly in advance. This model is built at the early stages of the software development. Fig 2.7 shows the process of prototyping

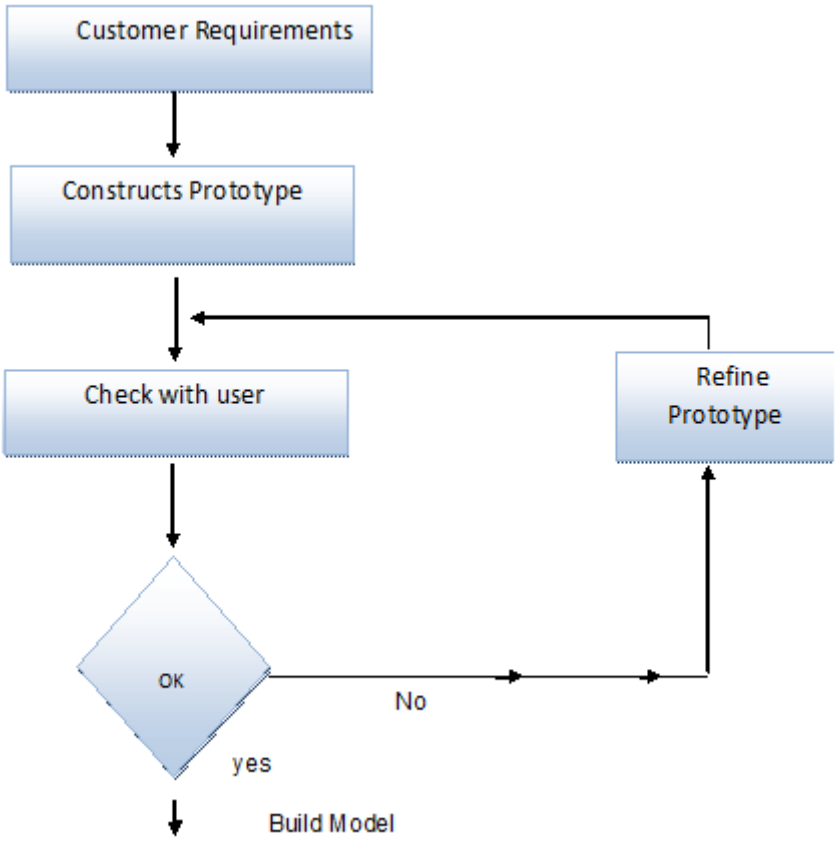


Figure: 2.7 Process of prototyping

Advantages of prototype model:

- Clear identification of requirements by developer
- Loop holes or drawbacks are indentified
- Faster system development
- Development & maintenance effort are reduced

Disadvantage of Prototype Model:

- Use of an inappropriate programming language due to quick building which later difficult or may be forgot by the developer to replace.
- Customer may visualize lack of good quality in a quickly built model

Q.6: Explain RAD Model/Rapid Application Development Model.

Ans: RAD model is also a speedy method of model development.

- Rapid Application development methodology is built with the time constraint of 2 to 3 months (for all components or functions)
- This model is an incremental model combining waterfall model with component based development model. He can also be said as modified version of waterfall model.
- Its development time is less including the use of reusable components.
- It is a high speed and fast processing model.
- It is majorly used in business oriented applications
- Here different & independent RAD frames are responsible for different modules and at last all the developed functions are integrated to form a complete system. Five phases of this model are shown in fig 2.8.

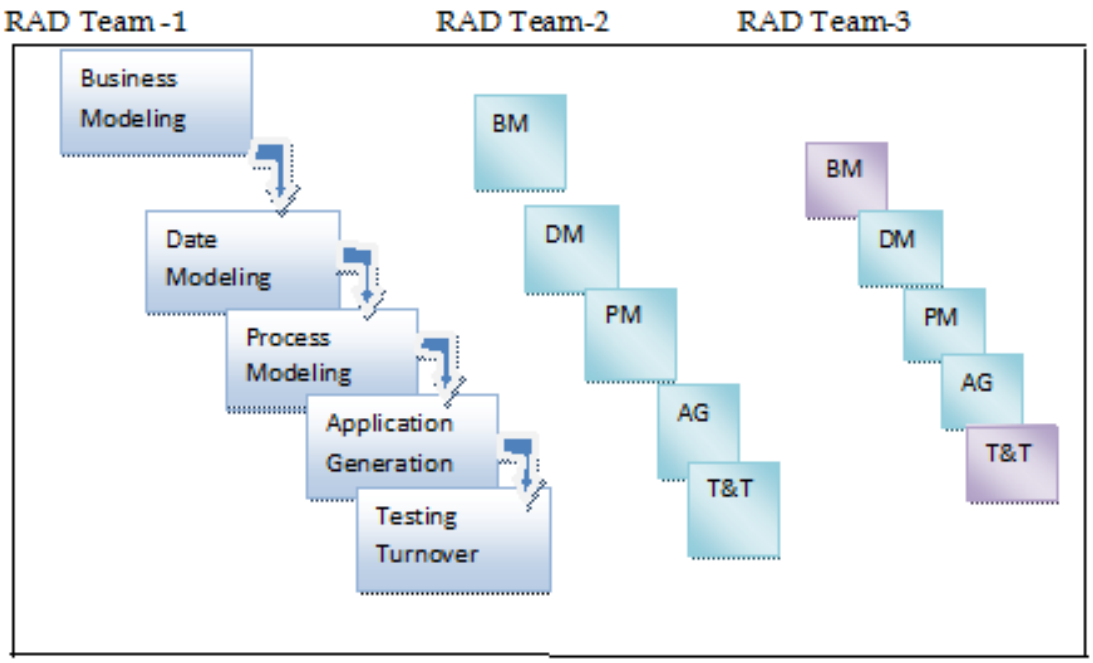


Figure: 2.8 RAD Model

Phases of RAD Model

Phase 1 Business Modeling (BM):
Information flow is defined & described with the description of source & destination of the information.

Phase 2 Data modeling (DM): Here the information flow is refined. For this, the information flow is partitioned into different data objects (having their own attributes). The relationship & connectivity is established between these data objects.

Phase 3 Process Modeling (PM): Data objects are combined to provide the information flow of each business function. Documents & reports are prepared related to the date manipulation of the date objects.

Phase 4 Application Generation (AG): Product is build will the help of new components & the already exiting components. The now built new components also serve as the receivable components for future purpose.

Phase 5 Testing & Turnover:
Unit testing is done for new added components as older components are already tested. After that integrated system is tested and the system is found satisfactory as per the user's requirements, delivered to the user or customer.

Advantages of RAD Model:
- Good for business- oriented application
- Less time is required for development
- Independent transaction for separate modules

Disadvantages of RAD Model:
- Time bounds or constraints should be met, otherwise useless for particular business requirements.
- Not suitable for all types of applications
- Sufficient manpower requirement for separate RAD teams
- Risk handling is absent in this development. Therefore, RAD Model is not suitable for the applications where the technical risk is high
- High interoperability between new & old technologies is required

Q.7 Explain Evolutionary Process Model

Ans: Evolutionary Process models are of four types:
1. Incremental
2. Spiral
3. WINWIN Spiral
4. Concurrent development model

- Evolutionary model expands the development stages in an incremental manner i.e. all the phases are developed one-by-one to incorporate all the well understood requirements.
- Evolutionary models are iterative in nature
- They are meant to develop increasingly more complete versions of the software
- It basically combines waterfall & prototype model

An evolutionary process model can be described as shown in Fig 2.8.

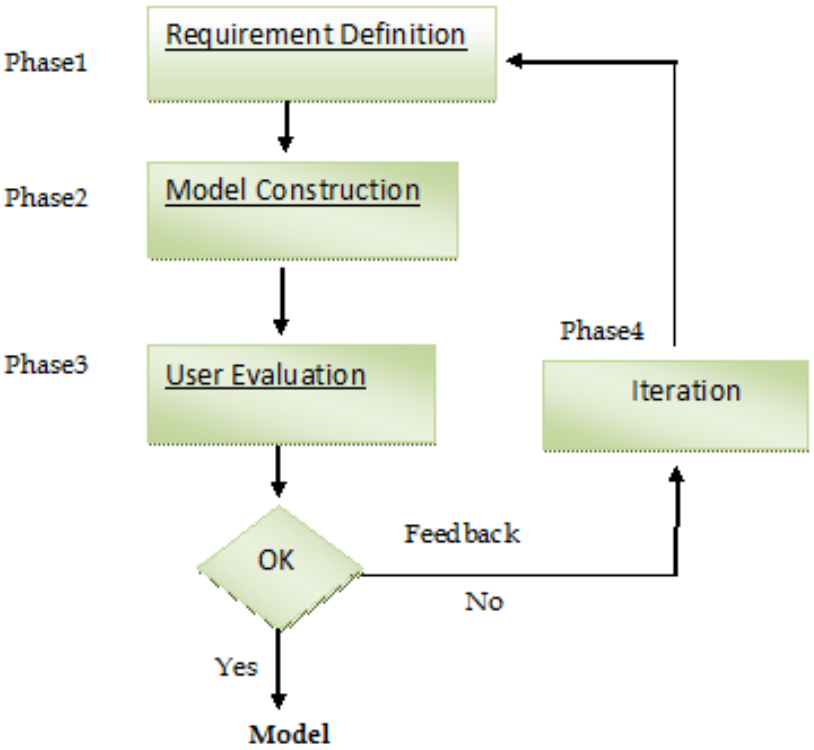


Figure: 2.8 Phases of Evolutionary Process Model

Q.8 Explain Incremental Model

Ans: Incremental model is:
-Introduced by "Mills"
-Combination of linear sequential & prototype model

-Each increment or interaction adds some functional capability to the software until the full software is implemented. The incremental model is shown in fig. 2.9.

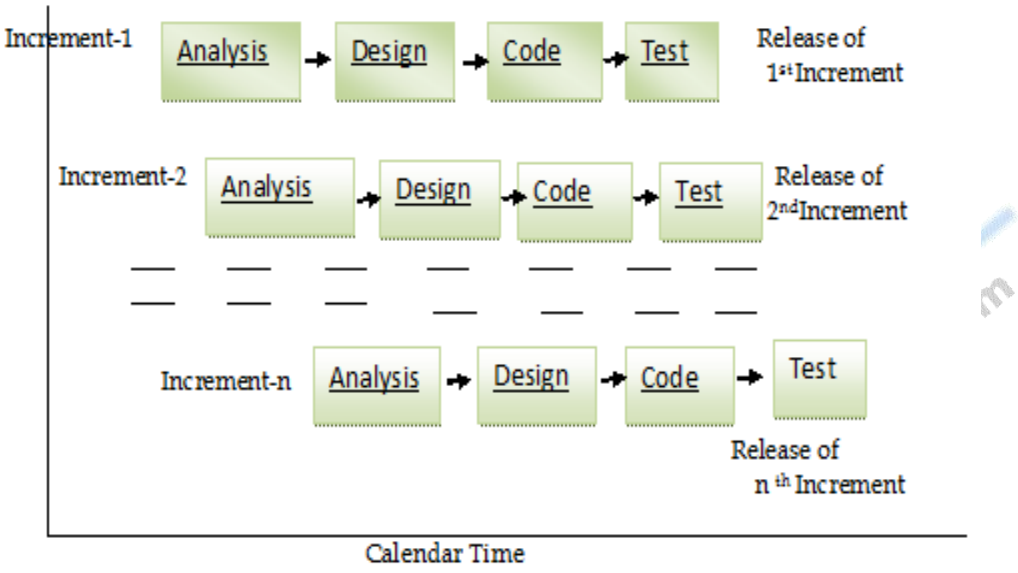


Figure: 2.9 Incremental Model

- With the release of each increment, experience is gained by developer as well as user.
- User can delay his decisions until he don't experiences the software.
- In the initial stages, user specifies his important & well known requirement. The release of first increment is also known as core product. This product is practically used & tested by the user for further clarification of their requirements. This addition of new requirements helps in building of the next increment with more capabilities. This process continues in the iteration mode until the nth iteration completes .At this stage the complete software is developed.

Advantage of Incremental Model

- Less Human resources are required at the beginning and as the number of increment increases, more members can be added to the project.

- Resource management could be done efficiently as if a resource which is felt to be included in an increment, if not available could be added in later increments.
- Regular and progressive evaluation of the new increments one after another (by the customer)
- User's customer gains experience & knowledge about the product with the release of increments.
- Lesser Project failure risk as a product accompanied with constant exposure to the user satisfaction & evaluation is revised with every increment.

Disadvantages of Incremental model

- Size of the increment should be optimum, otherwise small size could be a time consuming development and large one could be too complex.
- Categorization of best & most important requirements by the customers is difficult.

Ques.9: Explain Spiral Model

Ans: Spiral Model was introduced by " Boehm". It is:

- Combination of controlled & systematic aspects of linear sequential & prototype model.
- Introduction of an important phase i.e. phase of risk analysis
- This model is represented in the form of spirals where each spiral represents the various phases of the development. The basic structure of the spiral model is shown in fig 2.10.

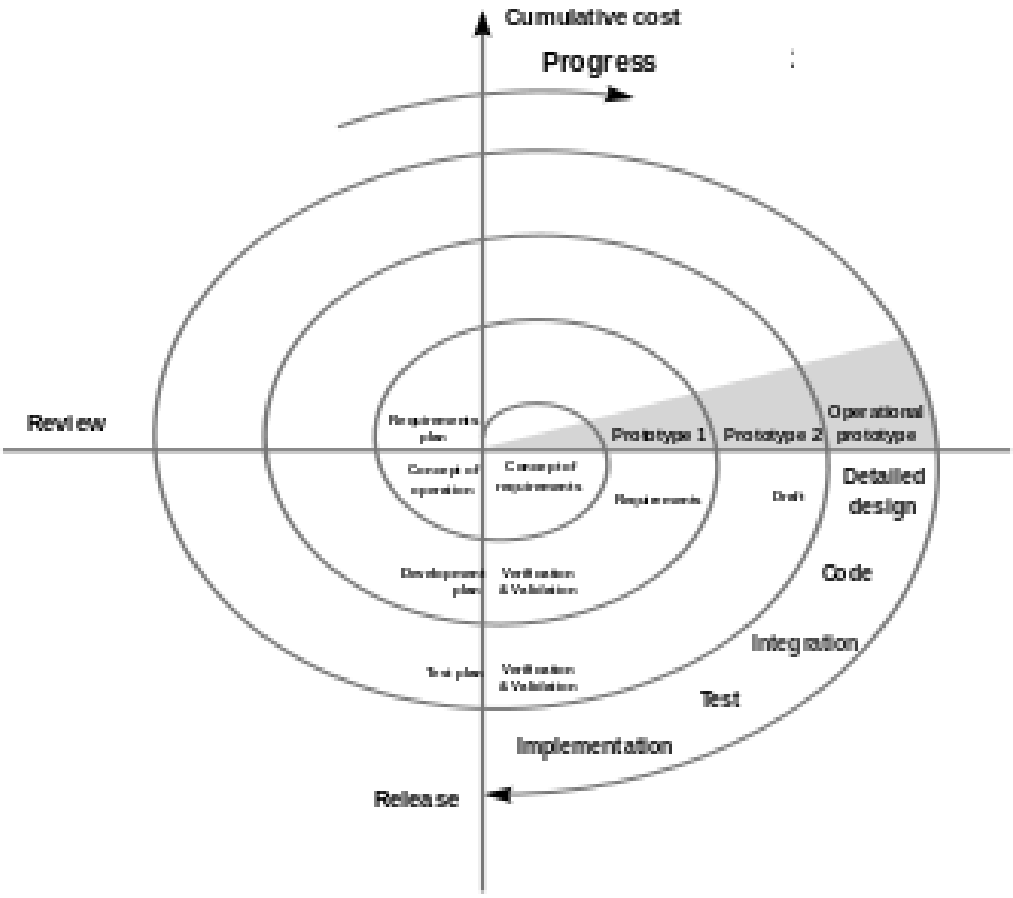


Figure 2.10: Spiral Model

A spiral model consists of four loops:
The process begins at the center of the spiral in clockwise direction.

- The first loop represents the system product concept.
- The second loop represents the development
- The third loop represents the testing
- The fourth loop represents the maintenance

Product concept:- Includes feasibility study & produces product specification
Product development:- Includes software building
Product testing:- Includes testing of the product

Product Maintenance:- Includes the maintenance of the product where the changes can be done in the controlled manner

A spiral model is portioned into 6 regions :

Region1. Customer Communication: - It is required to establish effective communication between software developer & customer.

Region2. Planning: - It is required to define resources, timelines and other projects related information.

Region3. Risk Analysis:- It is required to asses both technical & management risks.

Region4. Engineering: - It is required to build one or more presentations of the application and the best one is selected.

Region5. Construction & delivery: - It is required to construct, test, install and provide user support like user manuals and training.

Region6. Customer Evaluation: It is required to obtain customer feedback and evaluation of the software representation.

Advantages of the Spiral Model

- Software remains active throughout its life
- Software in maintainable
- Realistic Approach
- Reduced Risk

Disadvantage of Spiral Model

- It is difficult to satisfy the customer during evolution
- Risk analyst is required
- Slow development
- High monetary cost
- Requires more efforts for implementation

Q.10: Give some examples of risk in development.

Ans: In risk analysis, any raised risk should be handled regularly time-to-time. Risk minimization is an important aim of risk analysis. Few Examples of risk are:-

- If the team members leave the project during development, then the human resources risk will become high & it must be handled immediately by the risk analyst
- During a long development, the user requirements are rejected
- A competitor develops the same project
- Date lines not met
- Software performs too slow
- Software requires larger memory space than expected
- User changes requirements

Q.11 What is WINWIN spiral model?

Ans: It is an advancement of the spiral model. In spiral model, there is a simple communication between the developer & the customer related to the problem solution and there is no dealing between them, whereas in WINWIN spiral model, a process of negotiation or dealing takes place between customer & developer. The result of such negotiation is that, the customer wins by getting the software as per his requirements and the developer wins by getting the proper cost and schedule constraint. Thus the customer & developer both wins by getting their requirement fulfilled and get satisfy.

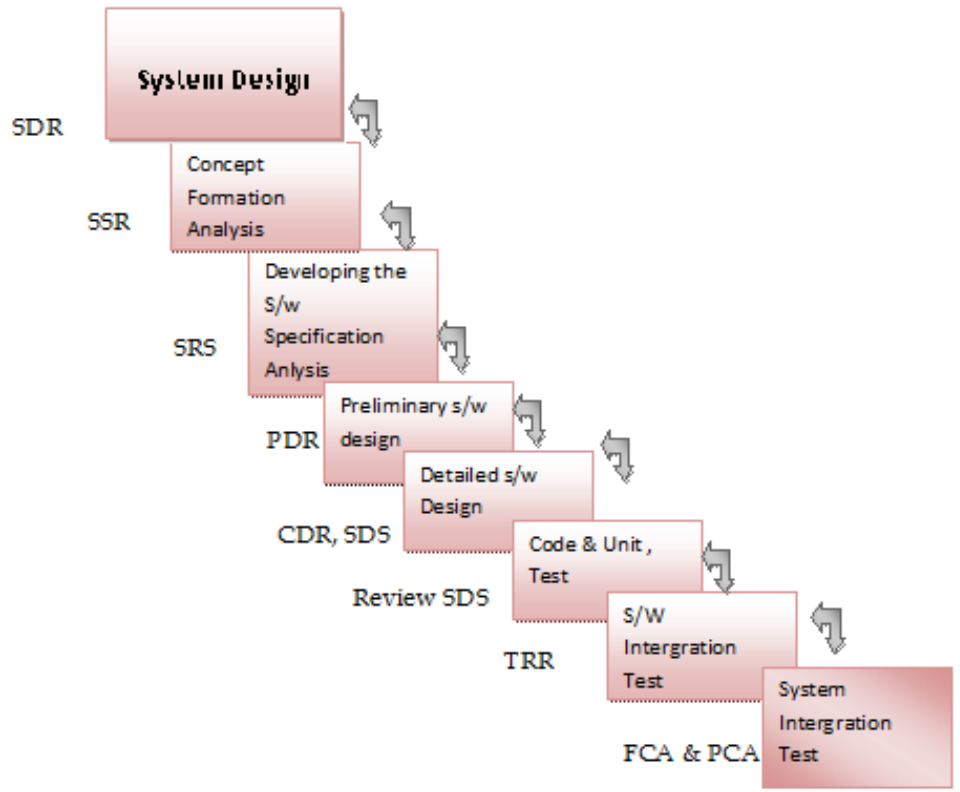
Q.12 What is Concurrent developmental model?

Ans: This is also known as Concurrent Engineering model.

- Here all activities of development are done simultaneously in parallel.
- For each activity, a state transition diagram is made, where all events of an activity are shown as states & there should be transitions between them.
- Provides an accurate idea about the states of the project.

Q.13: What is DOD model?

Ans: DOD Model is Department of Defense Model. It is based on the waterfall model. The DOD model is shown in Fig. 2.11.



SDR –System Design Review, SSR – System Software Review, SRS – Software Requirement Specification, PDR-Preliminary Design Review, CDR-Critical Design Review, SDS- Software Design Specification, TRR- Test Readline Review, FCA-Functional Configuration Audit, PCA-Physical Configuration Audit

Figure: 2.11 DOD Model

- The Main concept in DOD Model is that many of the phases are divided into its sub phases, so that all the phase work can be performed adequately & efficiently .Also a review at the end of each phase is conducted to evaluate the work done previously.
- Here, System design is kept as the primary phase of this model
- Software design phase is divided into preliminary software design phase & detailed software design
- The integration & test phase is considered separately for software & system.

Advantage of DOD:

- Well documented
- Satisfaction of requirements
- Elaborated process activities

Disadvantages of DOD:

- No user involvement
- Expensive
- No iteration
- Improper for development of expert system

Q.14: What is NASA model?

Ans: NASA Model is also known as NASA SALC (NASA Software Acquisition Life Cycle)

- Has well developed development life cycle
- Used for layer scale systems
- Structure similar to waterfall model

Project Management

Q.1 What is Software Project Management?

Ans: Software project management is the mean by which an orderly control process can be imposed on the software development process to ensure software quality along with the monitoring of the process and then controlling the wrong activities. Software project management is a crucial difficult task in case where (i) software execution is late (ii) over budget (iii) software fails to meet the user requirements.

Q.2 What are the basic project management process activities?

Ans: The basic project management process activities are:

1- Proposal Writing
2- Project Planning & scheduling
3- Project Costing
4- Project monitoring & reviews
5- Personal Selection & Evaluation
6- Report Writing & Presentations
7- Quality Management
8- Configuration Management

1- **Proposal Writing:**
It includes:
- Description of main objectives of the project
- How the objectives will be carried out and fulfilled
- Cost & schedule estimates

2- **Project Planning & Scheduling:**
Planning includes:
- Identification of activities
- Milestones (reports ,manual) for the management
- Deliverables for the customer

Project scheduling includes:
- Division of project into separate activities
- Time Judgment for completion of each activity

3- **Project Costing :**
It Includes:
- Estimation of the total cost of the projects

4- **Project Monitoring & Reviews :**

Monitoring is a continuing activity and includes:

- The progress of the project is compared regularly with the planned time schedule & costing (could be done with daily informal discussions or formal meetings

Review includes:

- The review of overall progress of the technical development of the project is done regularly.

5- **Personal Selection Evaluation :**

It includes:

- Selection of skilled & experienced staff for the project
- Regular evaluation of the performance of staff.
- Inexperienced staff may be trained.

6- **Report writing Presentation :**

- Report of project is briefly documented to present before the client & contractor.

7- **Quality Management**

It includes:

- Quality Assurance
- Quality planning
- Quality control

8- **Configuration Management :**

Configuration is a set of activities designed to control change by identifying the work products that are likely to change, establishing relationships among them, defining mechanisms for managing different version of these products, controlling & the changes imposed auditing & reporting on the changes made.

It includes:

- Identification of the work products.
- Managing the products & controlling, auditing, reporting the changes.

Q.3 What are the four P's of SPM?

Ans: Four "p" of Software project Management are:

- **People:** The People Factor is so important that Software Engineering institution has developed a People Management Capability Maturity Model. (PM-CMM). PM-CMM Defines the key practice areas for Software People in: Recruiting, Selection, Performance Management, Training, Compensation, career Development, Organization and work Design, Team Culture Development.

- **Product:** Objectives & scope of the required product should be established
- **Process:** A Software process provides the framework of development
- **Project:** Conduction of planned & controlled software project management is required to manage complexity

Q.4 What is Project Planning?

Ans: Project planning is included with in the project management activities. It involves reviewing datelines and use of the project completion schedules like Gantt Charts. These schedules help to plan and report for the project progress.

The inputs of the project planning phase: Project Charter and the Concept Proposal.
The outputs of the Project Planning phase: Project Requirements, the Project Schedule, and the Project Management Plan

Project planning includes the following steps:

1. Definition of project scope
2. Determination of appropriate methods for completing the project
3. Formation of Work Breakdown Structure (WBS) including various tasks and their duration.
4. Designing of activity network diagram to define the logical dependencies between tasks and identification of the critical path. Float/slack time in the schedule can be calculated using project management software.[
5. Estimation of total project cost by estimating necessary resources with cost for each activity

Q. 5 What is a project plan?

Ans: The software project plan is a relatively brief document that is addressed to a diverse audience.

This document majorly includes:
- Communication scope
- Resources, staffing, and customer description
- Definition of risks and risk management techniques
- Definition of cost and schedule for management review
- Provide an overall approach to software development

- Outline for techniques to ensure quality
-Process of change management

Q.6 What is Software project scheduling?

Ans: Software project scheduling is an activity that distributes estimated effort across the planned project duration by allocating the effort to specific software engineering tasks. Here, an outlined schedule is developed. And then, a detailed schedule is redefined for each entry in the macroscopic schedule.

Basic tasks and principles for software project scheduling are:

- Division of tasks or Compartmentalization
- Identification of Interdependency
- Time allocation
- Effort allocation & validation
- Defined responsibilities,
- Defined outcomes & milestones

Q.7 What are the methods of Project Scheduling?

Ans: Two project scheduling methods:
- Program Evaluation and Review Technique (PERT)
- Critical Path Method (CPM)

Both methods are driven by information developed in earlier project planning activities and include:
- Estimates of effort
- A decomposition of product function
- The selection of the appropriate process model
- The selection of project type and task set

Q.8 What is Risk Management?

Ans: Risk Management can be defined as a logical process to identify and analyze the appropriate methods for handling and monitoring exposures to loss or risk. It could be said as that, the risk management deals with systematic identification of an exposure to the risk of loss and making decisions on the best methods for handling these exposures to minimize losses.

Software Requirements Analysis

Q.1 What are Functional and non-functional requirements?

Ans: Functional requirements are the statements of services the system should provide, how the system should react to particular inputs and how the system should behave in particular situations. Functional requirements are the product capabilities, or things that a product must do for its users. Functional requirements define how software behaves to meet user needs.

Non-Functional Requirements are the constraints on the services or functions offered by the system such as timing constraints, constraints on the development process, standards, etc. Non-Functional Requirements in Software Engineering presents a systematic approach to `building quality into' software systems.

Q.2 What are User requirements?

Ans: User requirements are the statements in natural language with diagrammatic representations of the services, the system provides and its operational constraints.

Q.3 What are System requirements?

Ans: System requirements are the structured document specification containing detailed descriptions of the system's functions, services and operational constraints.

Q.4 What are Interface specification?

Ans: Interface specification includes the description of interfaces of the system and its subsystems. Such interface specification of subsystems allows independent development of the different subsystems. Interfaces may be defined as abstract data types or object classes.

Q.5 What is software requirement document?

Ans: The requirements document is the formal and official statement of what is required of the system developers. It includes both a definition of user

requirements and a specification of the system requirements; it specifies "what" the system should do rather than "how" it should do.

Requirement Engineering Process

Q.1 What do you mean by Requirement Engineering Process.

Ans: The processes used for Requirement Engineering mainly depend on the application domain, the people involved and the organization involved and responsible for developing the requirements. Also, there are a number of general activities common to all processes like, Requirements elicitation; Requirements analysis; Requirements validation; Requirements management. Figure 5.1 explains the Requirement Engineering Process.

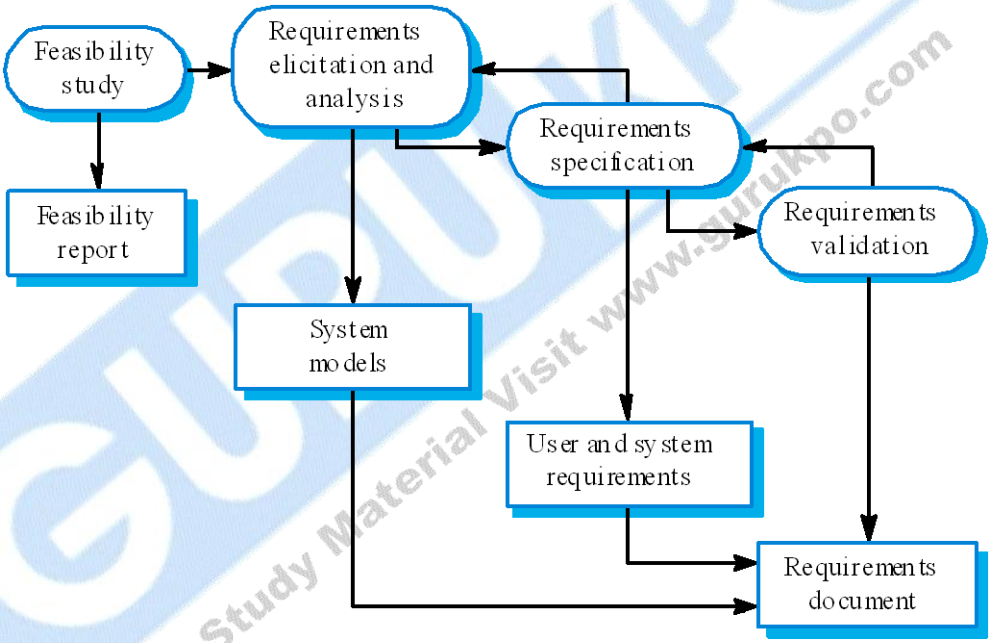


Figure 5.1 Requirement Engineering Process

Q.2 What do you mean by feasibility study?

Ans: A feasibility study decides whether or not the proposed system is fruitful to develop. Feasibility study checks: If the system contributes to organizational objectives; If the system can be engineered using current technology and within budget; If the system can be integrated with other systems that are used.

Three types of feasibility studies are:

1. Technical feasibility
2. Operational Feasibility
3. Economical Feasibility

Q.3 What do you mean by Requirement elicitation and analysis?

Ans: Requirement elicitation is also known as requirements discovery. It involves:
- Technical staff working with customers to find out about the application domain,
- The services that the system should provide and the system's operational constraints.
- May involve send-users, managers, engineers involved in maintenance, domain experts, trade unions, etc., called as stakeholders.

Q.4 What are the Problems of requirements analysis?

Ans: Few problems of requirements analysis are:
-Stakeholders don't know what they really want.
-Stakeholders express requirements in their own terms.
-Different stakeholders may have conflicting requirements.
-Organizational and political factors may influence the system requirements.
-The requirements change during the analysis process.
-New stakeholders may emerge and the business environment change.

Q.5 What are viewpoints?

Ans: Viewpoints are the way of structuring the requirements to represent the viewpoints of different stakeholders.

Q.6 What are the types of viewpoints?

Ans: Three Types of viewpoints are:

Interactor viewpoints: Here, People or other systems interact directly with the system. In an ATM, the customer's and the account database are the interactor viewpoints.
Indirect viewpoints: Here, Stakeholders who do not use the system themselves but influence the requirements. In an ATM, management and security staff are indirect viewpoints.
Domain viewpoints: Here, domain characteristics and constraints that influence the requirements. In an ATM, standards for inter-bank communication.

Q.7 What do you mean by Requirements validation?

Ans: Requirement Validation is the assurance done against the Software Requirement Specification (SRS) document. It verifies that the software being developed implements all the requirements specified in the SRS document. It helps in finding that, "Are we building the right product?".

Q.8 What do you mean by Requirements management?

Ans: Requirements management is the process of documenting, analyzing, tracing, prioritizing and agreeing on requirements of system development and then controlling change and communicating to relevant stakeholders. It is a continuous process throughout a project. At each stage in a development process, there are key requirements management activities and methods like Investigation, Feasibility, Design, Construction, Test, and Release of the system.

Software System Specifications

Q.1 What is Risk-driven specification?

Ans: Critical systems specifications are risk-driven, as risks create a threat to the system. This approach has been widely used in safety and security-critical systems. The aim of the specification process should be to understand the risks (safety, security, etc.) faced by the system and to define requirements that reduce these risks.

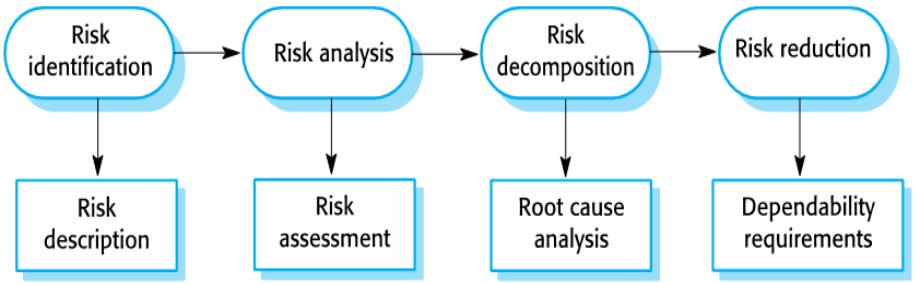


Figure 6.1 Risk Driven Specification

Stages of risk-based analysis:

- Risk identification
  - Identify potential risks that may arise.
- Risk analysis and classification
  - Assess the seriousness of each risk.
- Risk decomposition
  - Decompose risks to discover their potential root causes.
- Risk reduction assessment
  - Define how each risk must be taken into eliminated or reduced when the system is designed.

Q.2 What are Safety specifications?

Ans: Safety specifications are:

- Identifying protection requirements that ensure that system failures do not cause injury or death or environmental damage.
- Risk identification is equivalent to Hazard identification
- Risk analysis is equivalent to Hazard assessment
- Risk decomposition is equivalent to Hazard analysis
- Risk reduction is equivalent to safety requirements specification

Hazard identification

- Identify the hazards that may threaten the system.
- Hazard identification may be based on different types of hazard:
  - Physical hazards
  - Electrical hazards
  - Biological hazards
  - Service failure hazards
  - Etc.

Hazard assessment

- Estimate the hazard probability and the hazard severity.
- It is not normally possible to do this precisely so relative values are used such as 'unlikely', 'rare', 'very high', etc.
- The aim is to make sure that the system can handle hazards that are likely to arise or that have high severity.

Hazard analysis

- Concerned with discovering the root causes of risks in a particular system.
- Techniques have been mostly derived from safety-critical systems and can be
  - Inductive, bottom-up techniques. Start with a proposed system failure and assess the hazards that could arise from that failure;
  - Deductive, top-down techniques. Start with a hazard and deduce what the causes of this could be.

Risk reduction

- The aim of this process is to identify dependability requirements that specify how the risks should be managed and ensure that accidents/incidents do not arise.
- Risk reduction strategies
  - Risk avoidance;
  - Risk detection and removal;
  - Damage limitation.

Software Metrics and Measures

Q.1 What do you mean by Software Metrics and Measures?

Ans: Metrics provide the scale for quantifying qualities. For example, in software project the length is seen in terms of size metric of Software, then the unit of measurement of size is LOC (Line of Code).

Metrics provide a quantification of some property, whereas measurements provide the, actual value for the metrics. Software metric is a measure of some property of a piece of software or its specification.

Types of Metrics:

Indirect Metric: If metric can't be measured directly eg: Reliability of the software, that is estimated from other possible measurements.
Direct Metric: Metric that can be measured directly eg: Size

One way of Categorization of Software Metric

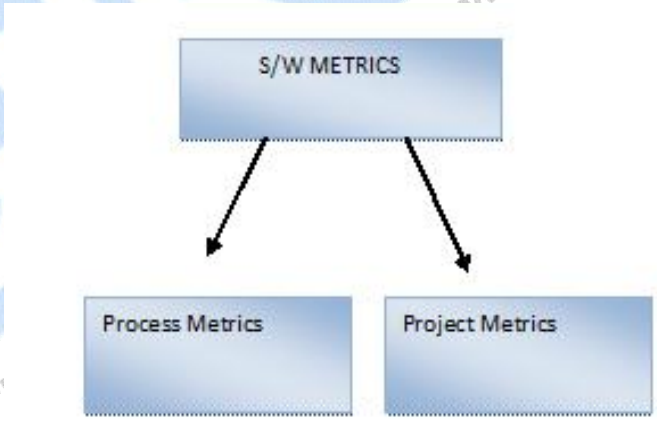


Figure 7.1 Categories of Metrics

Another way of Categorization of Software Metric

- Size-oriented Metrics

- Function-Oriented Metrics
- Human-Oriented Metrics
- Productivity Metrics
- Quality Metrics
- Technical Metrics

Q.2 What are process Metrics ?

Ans: Process metrics are used to check the progress of project development. These metrics leads to Software process improvement the steps of the software process improvement are as follows:

1- Specific attributes of process are measured
2- This results in the development of various metrics
3- Execution of these metrics will provide indicators
4- These indicators will improve the process

Eg: - Software measured in terms of reliability if it does not give satisfactory results, then improvement in this part is required.

Q.3 What is Project metric?

Ans: Project metrics are used by the project manager and a software team to check the project progress and the activities related to the project. Data from past projects are used to collect the various metrics (Time and Cost) then these metric and their estimates act as the base of the new software. As the project proceeds, the project manager will check its progress regularly time to time and compare the effort, time cost, with the estimated ones.

Q.4 What do you mean by Measuring Software and Measurement?

Ans: In the field of Engineering, it is easy to measure the power consumption, but in software development measure it is difficult to find that what to measure & how to evaluate measurements. As we know that Software measurement is required for the following reasons:

- To indicate the quality of the product
- To assess the productivity of the people who produce the product.
- To assess the advantage like productivity & quality derived from new software engineering methods & tools.
- To form a baseline for estimation
- To justify requests for new tools or additional training.

Categorization of Measurement:

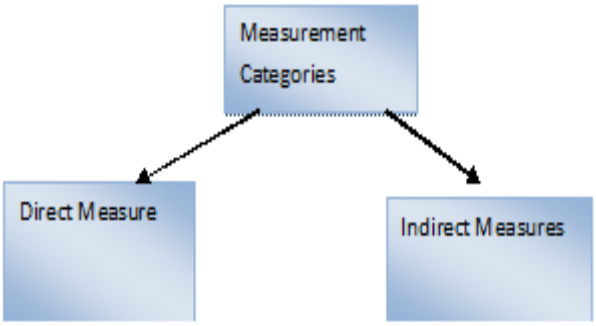


Figure 7.2 Measurement Categories

Direct Measures: Direct measures are the one which are measured directly from the software project itself. Eg: lines of code (LOC), memory size, execution speed.
Indirect Measures: These measures are not measured directly Eg: complexity reliability , maintainability.

Q.5 What are the various empirical estimation models.

Ans: An empirical estimation model for a software is used to predict effort as a function of Lines of Code (LOC) or Function Point (FP). "Basili" has described four types of resource Models:

1- Static single –variable models (ex-COCOMO)
2- Static Multivariable models
3- Dynamic Multivariable models
4- Theoretical Models

1- **Static Single Variable Model**:- The equation for this resource model is:

$$\text{Resource} = C_1 \times (\text{Estimated Characteristic})\ C_2$$

Where, resource= effort/project duration/staff size/pages of documentation
C_1 and C_2 = Constants derived from data collected from past projects

& Estimated characteristic = LOC/Efforts/Any Characteristics

2- **Static multivariable model**: These models use the historical data to derive empirical relationship. These model use the equation:

Resource = $C_{11}e_1$+ $C_{21}e_2$ +----------

Where, e_i = i^{th} software characteristics
C_1, C_2= Constts.
Eg. Of such model is Intermediate COCOMO

3- **Dynamic Multivariable Model** : Resource is defined as a function of time allocating some % of effort to each step in software engineering process.

4- **Theoretical Model:** Putnam is theoretical dynamic multivariable model. " Result is a resource expenditure curve".

Q.6 What is Risk Management?

Ans: Risk Management can be defined as: "A logical process for identifying and analyzing leading to appropriate methods for handling and monitoring exposures to loss".
Risk management deals with:

- Systematic identification of an exposure to the risk of loss, &
- Making decisions on the best methods for handling these exposures to minimize losses

Q.7 What is Risk Analysis with Risk Identification and Projection?

Ans: Risk Analysis includes:

- Risk identification
- Risk projection
  - impact of risks/likelihood of risk actually happening
- Risk assessment
  - what will change if risk becomes problem
- Risk management

Q.8 What is Risk Identification?

Ans: ds

Q.9 What is Risk Projection?

Ans: Risk Projection also called risk estimation, attempts to rate each risk on the basis of Probability and Consequences.

The risk drivers affecting each risk component are:
- classified according to their impact category
- potential consequences of each undetected software fault or unachieved project outcome are described

Q.10 Explain RMMM? Give a brief overview of RMMM plan.

Ans: RMMM is Risk Mitigation, Monitoring, and Management
An effective strategy must consider three issues:
- risk mitigation/reduction/avoidance,
- risk monitoring, and
- risk management and contingency(emergency) planning

- **Risk Mitigation**
  - proactive planning for risk avoidance
- **Risk Monitoring**
  - assessing whether predicted risks occur or not
  - ensuring risk aversion steps are being properly applied
  - collect information for future risk analysis
  - determining which risks caused which problems
- **Risk Management**
  - contingency planning
  - actions to be taken in the event that mitigation steps have failed and the risk has become a live problem

Q.11 Give a brief overview of RMMM plan.

Ans: **RMMM Plan:**

1. Introduction
- 1.1. Scope and Purpose of Document
- 1.2. Overview of major risks
- 1.3. Responsibilities
  - 1.3.1. Management
  - 1.3.2. Technical staff

2. Project Risk Table
- 2.1. Description of all risks above cut-off

2.2. Factors influencing probability and impact
3. Risk Mitigation, Monitoring, Management
3.1.1. Mitigation
3.1.1.1. General strategy
3.1.1.2. Specific steps to mitigate the risk
3.1.2. Monitoring
3.1.2.1. Factors to be monitored
3.1.2.2. Monitoring approach
3.1.3.Management
3.1.3.1. Contingency plan
3.1.3.2. Special considerations
4. RMMM Plan Iteration Schedule
5. Summary

Q.12 Explain Project Scheduling and Tracking?
Ans:
Software project scheduling is an activity that distributes estimated effort across the planed project duration by allocating the effort to specific software engineering tasks.
First, a macroscopic schedule is developed. Then, a detailed schedule is redefined for each entry in the macroscopic schedule. A Project schedule evolves over time.

Few guiding principles for software project scheduling are:

- Compartmentalization
- Interdependency
- Time allocation
- Effort allocation
- Effort validation
- Defined responsibilities
- Defined outcomes
- Defined milestones

Two well known project scheduling methods are:
- Program Evaluation and Review Technique (PERT)
- Critical Path Method (CPM)

Similarity between the two methods :
Both methods are driven by information developed in earlier project planning activities:

- Estimates of effort
- A decomposition of product function
- The selection of the appropriate process model
- The selection of project type and task set

Both methods allow a planner to do:
- determine the critical path
- time estimation
- calculate boundary times for each task

Boundary times:
- the earliest time and latest time to begin a task
- the earliest time and latest time to complete a task
- the total float.

Q.13 Explain Project Tracking?

Ans: The project schedule provides a road map for a software project manager. It defines the tasks and milestones.

Several ways to track a project schedule:

- Conducting periodic project status meeting
- Evaluating the review results in the software process
- Determine if formal project milestones have been accomplished
- Compare actual start date to planned start date for each task
- Informal meeting with practitioners

Application Systems and Design Issues

Q.1 Explain various application systems.

Ans:

a) Data processing systems- Computerized system that performs mathematical operations or manipulations on input-data to transform it into the output form like audio/video, graphic, numeric, or text as per the system user's information requirements.

b) Transaction processing systems- A transaction process system (TPS) is an information processing system for business transactions involving the collection, modification and retrieval of all transaction data. Characteristics of a TPS include performance, reliability and consistency.

c) **Event processing systems-** Event Processing systems deal with the task of processing one or many events with the goal of identifying the meaningful events within the event cloud.

d) Language **processing systems-**Such systems are concerned with the interactions between computers and human (natural) languages

Q.2 Explain User Interface Design.

Ans: User interfaces should be designed to match the skills, experience and expectations of its anticipated users. System users often judge a system by its interface rather than its functionality. A poorly designed interface can cause a user to make catastrophic errors. Poor user interface design is the reason why so many software systems are never used.

Q.3 What do you mean by user interface design process.

Ans: User Interface design is an iterative process involving close liaisons between users and designers. The process is explained in fig 8.1. The three core activities in this process are:

- **User analysis**. Understand what the users will do with the system;
- **System prototyping**. Develop a series of prototypes for experiment;
- **Interface evaluation**. Experiment with these prototypes with users.

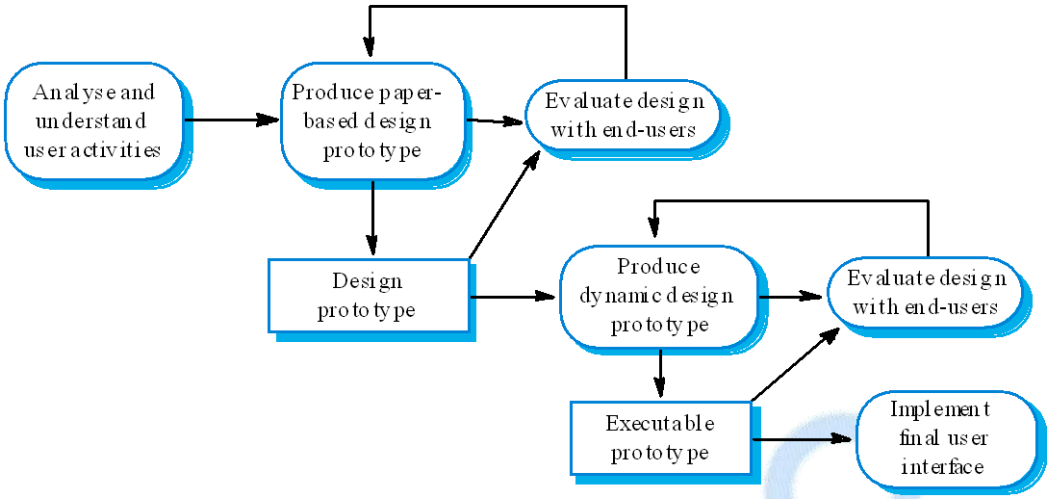


Figure 8.1 Interface Design Process

Q.4 What do you mean by User analysis.

Ans: In User Analysis, the analyst makes every effort to get the end user's mental model and the design model to converge by understanding

- The users themselves
- How these people use the system

Analysis techniques:

1)Task analysis- Models the steps involved in completing a task.
2)Interviewing and questionnaires-Asks the users about the work they do.
3)Ethnography-Observes the user at work.

Software Development Methods and Reuse

Q.1: Explain Rapid Software Development.

Ans: Due to rapid changes in requirements and business environments, the software requires rapid development and delivery. But is not often the most critical requirement for software systems. Businesses may be willing to accept lower quality software if rapid delivery of essential functionality is possible.

Characteristics of RAD processes

- The processes of specification, design and implementation are concurrent. There is no detailed specification and design documentation is minimised.
- The system is developed in a series of increments. End users evaluate each increment and make proposals for later increments.
- System user interfaces are usually developed using an interactive development system.

RAD environment tools

- Database programming language
- Interface generator
- Links to office applications
- Report generators

Q.2: What do you mean by Agile methods?

Ans: Agile methods are the methods which focus on the code rather than the design, based on an iterative approach to software development process and are intended to deliver working software rapidly to meet the rapid changing requirements. Agile methods are best suited to small and medium-sized systems or personal computer products.

Problems with agile methods

- It can be difficult to keep the interest of customers who are involved in the process.
- Team members may be unsuited to the intense involvement that characterizes agile methods.

- Prioritizing changes can be difficult where there are multiple stakeholders.
- Maintaining simplicity requires extra work.
- Contracts may be a problem as with other approaches to iterative development.

Q.3: What do you mean by Extreme programming?

Ans: Extreme Programming or XP takes an 'extreme' approach to iterative development. Here:

- New versions may be built several times per day;
- Increments are delivered to customers every 2 weeks;
- All tests must be run for every build and the build is only accepted if tests run successfully.

Q.4 What do you mean by Software prototyping?

Ans: A prototype is an initial version of a system used to demonstrate concepts and try out design options. A prototype can be used in:

- The requirements engineering process to help with requirements elicitation and validation;
- In design processes to explore options and develop a User Interface design;
- In the testing process to run back-to-back tests.

Benefits of prototyping

- Improved system usability.
- A closer match to users' real needs.
- Improved design quality.
- Improved maintainability.
- Reduced development effort.

Q.5 What do you mean by Software Reuse?

Ans: Software reuse is an act of developing new system through the usage of previously available components or other systems.

Q.6 What do you mean by Design Pattens?

Ans: A design pattern is a way of reusing abstract knowledge about a problem and its solution. A pattern is a description of the problem and the essence of its solution. It should be sufficiently abstract to be reused

in different settings. Patterns majorly rely on object characteristics such as inheritance and polymorphism.

Q.7 What do you mean by Generator-based reuse?

Ans: Program generators involve the reuse of standard patterns, algorithms and procedures. These are embedded in the generator and parameterized by user commands. A program is then automatically generated. Generator-based reuse is possible when domain abstractions and their mapping to executable code can be identified. A domain specific language is used to compose and control these abstractions.

Q.8 What do you mean by Application frameworks?

Ans: Application frameworks are collections of abstract and concrete classes that can be adapted and extended to create application systems.

Q.9 What do you mean by Application frameworks?

Ans: In Application framework, the whole of an application system may be reused either by incorporating it without change into other systems or by developing application families.

Q.10 What is Software Evolution?

Ans: It is the process of developing software initially, then repeatedly updating it for various reasons. With time, software systems, programs as well as applications, continue to develop. These changes will require new laws and theories to be created and justified. Innovations and improvements do increase unexpected form of software development.

Q.11 What are Lehman's Laws of Software Evolution?

Ans: Lehman's Laws of software evolution can be defined as:

- Continuing Change: Systems must continually be adapted to the changing environment, otherwise their utility will progressively decline.
- Increasing Complexity: The accidental and essential complexity grows as the system is evolved.

- Large Program Evolution
- Invariant Work-Rate
- Conservation of Familiarity: The long-term growth rate of the system declines.
- Continuing Growth: The functional capacity of the system must continually be improved over the system lifespan to maintain stakeholder satisfaction.
- Declining Quality: The quality of the system declines unless dedicated countermeasures are taken.
- Feedback System: Evolution processes are multi-level, multi-loop, multi-agent feedback systems.

Verification and Validation

Q.1 Explain Verification and Validation.

Ans: **Verification:** "Are we building the product right". The software should conform to its specification.
Validation: "Are we building the right product". The software should do what the user really requires.

Verification & Validation must be applied at each stage in the software process. Verification & Validation has two principal objectives:

- The discovery of defects in a system
- The assessment of whether or not the system is useful and useable in an operational situation.

Verification and validation should establish confidence that the software is fit for purpose, but this does NOT mean completely free of defects.

Q.2 What do you mean by Software inspections?

Ans: Software inspections are concerned with analysis of the static system representation to discover problems (static verification). It may be supplemented by tool-based document and code analysis.

- These involve people examining the source representation with the aim of discovering anomalies and defects.
- Software Inspections does not require execution of a system so may be used before implementation.
- They may be applied to any representation of the system (requirements, design, configuration data, test data, etc.).
- They have been shown to be an effective technique for discovering program errors.
- Inspections and testing are complementary and not opposing verification techniques. Both should be used during the V & V process.
- Inspections can check conformance with a specification but not conformance with the customer's real requirements.
- Inspections cannot check non-functional characteristics such as performance, usability, etc.

Q.3 What is automated static analysis?

Ans: Automated static analysis involves the static analysers. Static analysers are software tools for source text processing. They parse the program text and try to discover potentially erroneous conditions and bring these to the attention of the Verification and Validation team. They are very effective and helpful for inspections.

Q.4 What do you mean by Verification and formal methods.

Ans: Formal methods can be used when a mathematical specification of the system is produced. They are the ultimate static verification technique. They involve detailed mathematical analysis of the specification and may develop formal arguments that a program conforms to its mathematical specification.

Software Testing and Cost Estimation

Q.1 Explain System testing.

Ans: System Testing is the assurance of proper working or execution of the developed system from user's perspective using formal procedure.

System Testing Process can be defined in the following steps (as shown in fig 11.1):

a. **Function Testing**: the system must perform functions specified in the requirements.
b. **Performance Testing**: the system must satisfy security, precision, load and speed constrains specified in the requirements.
c. **Acceptance Testing**: customers try the system to make sure that the system built is the system they had requested.
d. **Installation Testing**: the software is deployed and tested in the production environment.

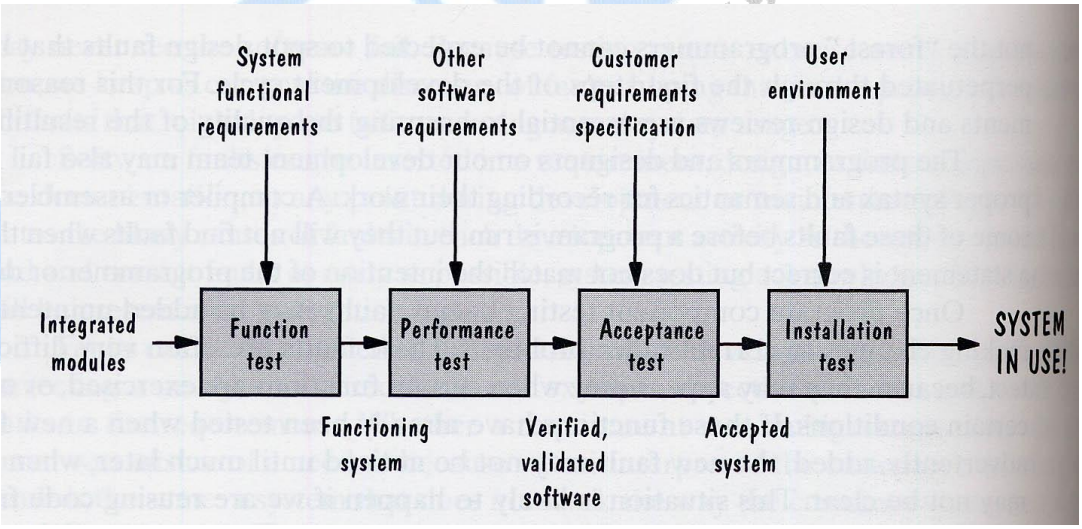


Figure 11.1: System Testing Process

Q.2 Explain Component testing.

Ans: Component testing is also known as Unit testing. It refers to tests that verify the functionality of a specific section of code, usually at the

function level. In an object-oriented environment, this is usually at the class level, and the minimal unit tests include the constructors and destructors. These types of tests are usually written by developers as they work on code (white-box testing), to ensure that the specific function is working as expected. Unit testing alone cannot verify the functionality of a piece of software, but rather is used to assure that the building blocks of the software work independently of each other.

Q.3 Explain Test case design.

Ans: Test Case design includes the designing of test cases. A Test Case is a set of actions executed to verify a particular feature or functionality of your software application. A *test case* is a detailed procedure that fully tests a feature or an aspect of a feature. Whereas the test plan describes what to test, a test case describes how to perform a particular test. You need to develop a test case for each test listed in the test plan.

A test case includes:

- The purpose of the test.
- Special hardware requirements, such as a modem.
- Special software requirements, such as a tool.
- Specific setup or configuration requirements.
- A description of how to perform the test.
- The expected results or success criteria for the test.

Tips for writing test cases:

1. Test Cases need to be simple and transparent
2. Create Test Case with End User in mind
3. Avoid test case repetition
4. Do not Assume
5. Test Cases must be identifiable
6. Implement Testing Techniques
7. Peer Review

Q.4 What is Test automation?

Ans: In software testing, test automation is the use of special software (separate from the software being tested) to control the

execution of tests and the comparison of actual outcomes to predicted outcomes. Test automation can automate previous repetitive but necessary testing in a formalized testing process already in place, or add additional testing that would be difficult to perform manually.

Q.5 What do you mean by Software productivity?

Ans: In standard economic terms, productivity is the ratio between the amount of goods or services produced and the labor or expense that goes into producing them. Software productivity is the ratio between the amount of software produced to the labor and expense of producing it. Some of the methods of measuring software productivity are Function Point Analysis, Constructive Cost Modeling, and Cyclomatic Complexity.

Q.6 What is Algorithmic cost modeling.

Ans: It is formulaic approach based on historical cost information and which is generally based on the size of the software. In algorithmic cost modeling, cost is estimated as a mathematical function of product, project and process attributes whose values are estimated by project managers.

Effort = A x $Size^B$ x M

Where, A=organisation-dependent constant,
B= disproportionate effort for large projects
M=is a multiplier reflecting product, process and people attributes. (Most commonly used product attribute for cost estimation is code size)

Most models are basically similar but with different values for A, B and M

Q.7 What do you mean by Project duration and staffing?

Ans: Managers are supposed to estimate the calendar time required to complete a project and the points when staff will be required. Calendar time can be estimated using a COCOMO 2 formula

$TDEV = 3 \times (PM)^{(0.33+0.2*(B-1.01))}$

Where,

PM = effort computation

B = exponent computed (E.g. B =1 for the early prototyping model)

Staffing includes the following facts:

- The time requirement is independent of the number of people working on the project.
- Staff required can't be computed by diving the development time by the required schedule
- The number of people working on a project varies depending on the phase of the project
- The more people who work on the project, the more total effort is usually required
- A very rapid build-up of people often correlates with schedule slippage

Q.8 What is COCOMO?

Ans: COCOMO is Constructive Cost Model and it is an empirical model based on project experience. It is well-documented, 'independent' model which is not tied to a specific software vendor**.**

COCOMO Versions

1. COCOMO 81" or COCOMO I (Base Model)
2. Cocomo II (circa 2000) (Current Version)
3. Commercial take-offs
    - Costar (Softstarsystems.com)
    - Cost Xpert (CostXpert.com)

COCOMO I or COCOMO 81

- Proposed by B.W. Boehm in 1981
- Based on related data of 63 completed software projects.
- Developed for Cost Estimation i.e. for Person-Months and Development Time from size estimation
- It is a static model

COCOMO II MODEL

- Successor of COCOMO 81

- Developed at University of Southern California in 1995 under the leadership of Dr. Barry Boehm
- Based on 83 projects
- COCOMO II takes into account new development processes, increased flexibility in software development, need for decision making with incomplete information and new data about projects.
- Provides more support for estimating modern software development processes and an updated project database.
- The need for the new model was raised due to updating and enhancement in dev. technology like desktop development, code reusability and the use of off-the-shelf software components.
- COCOMO II models require sizing information. Three different sizing options are available as part of the model hierarchy i.e.
    - OP, object points,
    - FP, function points,
    - LOC, lines of source code.

Q.9 What are the types of COCOMO I or COCOCMO 81 Model?

Ans: Types of COCOMO-81 Model:

- Basic COCOMO Model
- Intermediate COCOMO Model
- Detailed COCOMO Model

Q.10 Explain Basic COCOMO Model.

Ans: Basic COCOMO is a Single Variable Static Model.
It computes software development effort (and cost) as a function of program size.

E = f(Program size)
Program size is expressed in estimated thousands of source lines of code (KLOC)

On the basis of development complexity Boehm described 3 categories or modes of a project–

- ORGANIC Projects
- SEMIDETACHED Projects
- EMBEDDED Projects

Organic projects
These are relatively small, simple SW projects with "good" experience "small" teams working with "less than rigid" requirements (Ex. Application programs like simple business systems, inventory systems, data processing systems, etc.) – (0-50 KLOC)
Semi-detached projects
An intermediate level (size and complexity) SOFTWARE projects with mixed experience team members and intermediate requirement rigidity. It can be mixture of organic and embedded software as well. (Ex : utility programs like compilers, linkers, complex inventory systems etc.)-(50-300 KLOC)
Embedded projects
System programs which are developed within tight HW, SW and operational constraints (Ex . Real time systems, embedded systems, flight control SW for aircraft etc.)-(Over 300 KLOC)

The Basic COCOMO equations:

Effort (E) = a_b (KLOC) exp (b_b)

Dev Time (D) = c_b (Effort) exp(d_b)
People required (P) = E / D
where, **KLOC** is the estimated no. of thousands of lines of code for project.
a_b, b_b, c_b and d_b are constants for each mode of software products. The values for a_b, b_b, c_b and d_b are as follows:

Software projects	a_b	b_b	c_b	d_b
Organic	2.4	1.05	2.5	0.38
Semi-detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

Basic COCOMO is good for quick estimate of software costs.
But, it does not account for :-

- differences in hardware constraints,
- personnel quality and experience,
- use of modern tools and techniques.

Q.11 Explain Intermediate COCOMO Model.

Ans: It is a Multi Variable Static Model. It computes Effort as function of program size and a set of 4 "cost drivers"

E = f (Program size, cost drivers)

where, "cost drivers" includes subjective assessment of product, hardware, personnel and project attributes

Set of four "cost drivers"

- Product attributes
- Hardware attributes
- Personnel attributes
- Project attributes

Cost Drivers with subsidiary attributes (total 15)

Product attributes

Required software reliability
Size of application database
Complexity of the product

Hardware attributes

Run-time performance constraints
Memory constraints
Volatility of the virtual machine environment
Required turnabout time

Personnel attributes

Analyst capability
Software engineering capability
Applications experience
Virtual machine experience
Programming language experience

Project attributes

Use of software tools
Application of software engineering methods
Required development schedule

Each of the 15 attributes receives a rating on a six-point scale that ranges from "very low" to "extra high".

An effort multiplier from the table shown next applies to the rating. The product of all effort multipliers results in an *effort adjustment factor (EAF)*. Typical values for EAF range from 0.9 to 1.4.

Cost Drivers	Ratings					
	Very Low	Low	Nominal	High	Very High	Extra High
Product attributes						
Required software reliability	0.75	0.88	1.00	1.15	1.40	
Size of application database		0.94	1.00	1.08	1.16	
Complexity of the product	0.70	0.85	1.00	1.15	1.30	1.65
Hardware attributes						
Run-time performance constraints			1.00	1.11	1.30	1.66
Memory constraints			1.00	1.06	1.21	1.56
Volatility of the virtual machine environment		0.87	1.00	1.15	1.30	
Required turnabout time		0.87	1.00	1.07	1.15	
Personnel attributes						
Analyst capability	1.46	1.19	1.00	0.86	0.71	
Applications experience	1.29	1.13	1.00	0.91	0.82	
Software engineer capability	1.42	1.17	1.00	0.86	0.70	
Virtual machine experience	1.21	1.10	1.00	0.90		
Programming language experience	1.14	1.07	1.00	0.95		
Project attributes						
Application of software engineering methods	1.24	1.10	1.00	0.91	0.82	
Use of software tools	1.24	1.10	1.00	0.91	0.83	
Required development schedule	1.23	1.08	1.00	1.04	1.10	

The Intermediate Cocomo formula :

$E = a(KLoC)^{(b)}.EAF$

Dev. Time $= c.(E)^{(d)}$

where $EAF = C_1 x\ C_2 x\ C_3 x\ C_4 x\ C_5 \ldots\ldots x\ C_{15}$

and the values for a, b, c and d are shown as:

Software project	a	b	c	d
Organic	3.2	1.05	2.5	0.38
Semi-detached	3.0	1.12	2.5	0.35
Embedded	2.8	1.20	2.5	0.32

Q.12 Explain Detailed COCOMO Model.

Ans: Detailed COCOMO incorporates all characteristics of the intermediate version with an assessment of the cost driver's impact on each step (analysis, design, etc.) of the software engineering process. The detailed model uses different efforts multipliers for each cost drivers attribute. The effort multipliers are Phase Sensitive.

Here,
E = f (program size, cost drivers)
Where cost drivers= f (phases)

The five phases of detailed COCOMO are:-
- plan and requirement
- system design
- detailed design
- module code and test
- integration and test

Q.13 What are the different types of COCOMO II Model?

Ans: The sub-models in COCOMO II are:

1. Application composition model
2. Early design model
3. Reuse model
4. Post-architecture model

Q.14 Explain Application composition model.

Ans: Application composition model is:

- Also called as Early Prototyping Model.
- Size estimation is based on object points.
- Then these object points are used to find the new object points
- Then effort is calculated by using the productivity.
- Supports prototyping projects and projects where there is extensive reuse
- Suitable for projects built with modern GUI-builder tools.
- Takes CASE (Computer Aided Software Engineering Environment) tool use into account.

- Object Point is an indirect software measure.
- To compute object Points, count the number of screens, reports and third Generation Language (3GL) components likely to be required to build the application.
  Formula is:-
  E = NOP / PROD
  where, E = Effort
  NOP=New Object Pts. or Application Pts.
  PROD = Productivity

Q.15 Explain Early design model.

Ans: In Early design model:

- Estimation is done using information available in the early stages of the project.
- Used when not much detail is known.
- This model uses function points.

Formula is:
$E = A \times Size^{B} \times M$
Where,

- M = PERS × RCPX × RUSE × PDIF × PREX × FCIL × SCED; these are the process drivers and values of these drivers ranges from 1 (for low) to 6 (very high)
- A = 2.5
- Size in KLOC
- B varies from 1.1 to 1.24 depending on novelty of the project, development flexibility, risk management approaches and the process maturity.

Multipliers:
Multipliers reflect the capability of the developers, the non-functional requirements, the familiarity with the development platform, etc.
RCPX - product reliability and complexity;
RUSE - the reuse required;
PDIF - platform difficulty;
PREX - personnel experience;
PERS - personnel capability;
SCED - required schedule;
FCIL-the team support facilities

Q.16 Explain Reuse model.

Ans: The Reuse Model takes into account black-box code that is reused without change and code that has to be adapted to integrate it with new code.
There are two versions:

- Black-box reuse where code is not modified. An effort estimate (PM) is computed.
- White-box reuse where code is modified. A size estimate equivalent to the number of lines of new source code is computed. This then adjusts the size estimate for new code.

For generated code:

E = (KLOC * AT/100)/ATPROD
Where,
AT is the percentage of code automatically generated.
ATPROD is the productivity of engineers in integrating this code.
When code has to be understood and integrated:
E = KLOC * (1-AT/100) * AAM
Where,

- AT is the percentage of code automatically generated.
- AAM is the adaptation adjustment multiplier computed from the costs of changing the reused code, the costs of understanding how to integrate the code and the costs of reuse decision making.

Q.16 Explain Post-Architecture model.

Ans: Post-Architecture Model:

- Is the Most detailed Estimation model
- Assumes that the system architecture is known and the information on the various cost drivers of the system is available.
- Uses LOC for size estimation
- Uses the same formula as the early design model but with 16 rather than 7 associated multipliers.

Q.17 Differentiate between COCOMO I and COCOCMO II Model.

Ans: Few differences between COCOMO I and COCOMO II are:

- COCOMO I provides point estimates of effort and schedule, but COCOMO II provides likely ranges of estimates.
- In COCOMO II, the estimation equation exponent is determined by five scale factors instead of three.
- Data points in COCOMO I: 63 and COCOMO II: 161.
- COCOMO II adjusts for software reuse and re-engineering but COCOMO I made little accommodation for these factors.

Q.18 Explain PUTNAM Estimation Model.

Ans: PUTNAM Model is:

- A theoretical multivariable dynamic model.
- Proposed by Lawrence Putnam in 1977
- Based on Data of around 4000 projects.
- Relationship between size and effort is non linear.
- It assumes a specific distribution of effort over the life cycle of software development.
- One of the distinguishing features of the Putnam model is that total effort decreases as the time to complete the project is extended.
- For large software projects: the Rayleigh –Norden curve relating the manpower and effort is as shown in fig. 11.1:

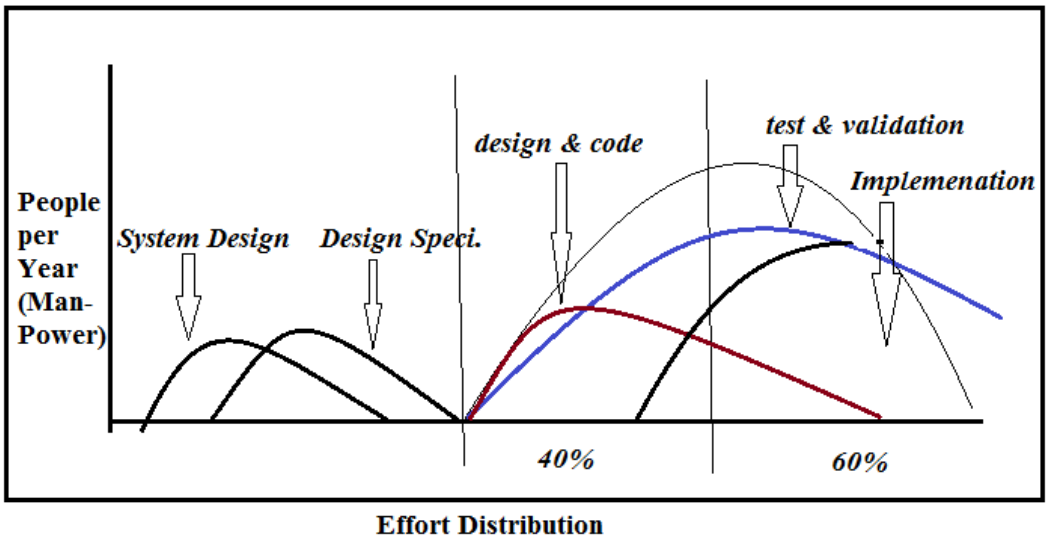


Figure 11.1 Rayleigh –Norden curve

Formula:

$$E = [LOC \times B^{0.33} / P]^3 \times (1/ t^4)$$

Where,

E =Effort

B=Special Skill Factor based on size

P=Productivity parameters

t =Project duration in months or years

Values of B,

B=0.39 for >=70 KLOC

B = 0.16 for 5-15 KLOC

Values of P (depends on technology & environment),

P=2,000 for Real time software

P=10,000 for system software and telecomm software

P=28,000 for business system software

Quality Management

Q.1 Explain process and product quality

Ans: Understanding, Modelling and Improving the Software Process is process improvement.

- Process quality and product quality are closely related
- A good process is usually required to produce a good product
- For manufactured goods, process is the principal quality determinant
- For design-based activity, other factors are also involved especially the capabilities of the designers

Principal product quality factors are as shown in fig 12.1.

Figure 12.1 Principal product quality factors

- For large projects with average capabilities, the development process determines product quality
- For small projects, the capabilities of the developers is the main determinant
- The development technology is particularly significant for small projects
- In all cases, if an unrealistic schedule is imposed then product quality will suffer

Q.2 What do you mean by Quality assurance and its standards?

Ans: Quality assurance is establishing organisational procedures and standards for quality.

Software Quality Assurance (SQA) Activities:

- Prepares an SQA plan for a project
- Participates in the development of the project's software process description
- Reviews software engineering activities to verify compliance with the defined software process
- Audits designated software work products to verify compliance with those defined as part of the software process
- Ensures that deviations in software work and work products are documented and handled according to a documented procedure
- Records any noncompliance and reports to senior management
- Coordinates the control and management of change
- Helps to collect and analyze software metrics

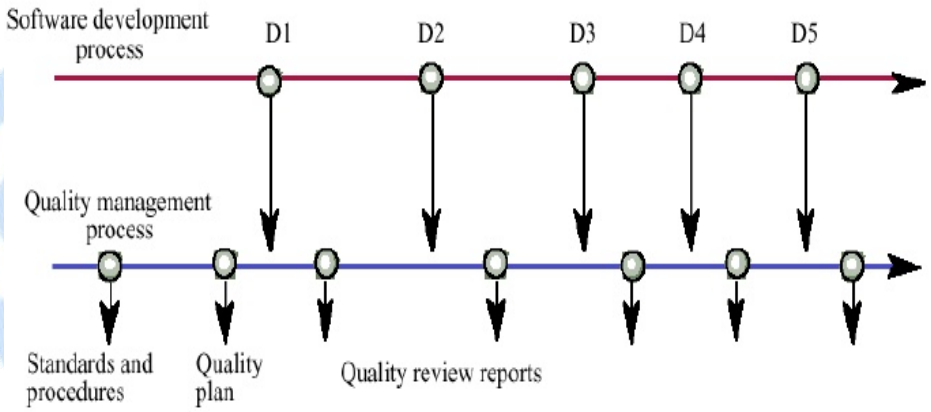


Figure 12.2 Principal product quality factors

Standards:

- Standards are the key to effective quality management
- They may be international, national, organizational or project standards

- Product standards define characteristics that all components should exhibit e.g. a common programming style
- Process standards define how the software process should be enacted

Importance of standards:
- Encapsulation of best practice- avoids repetition of past mistakes
- Framework for quality assurance process - it involves checking standard compliance
- Provide continuity - new staff can understand the organisation by understand the standards applied

Certification Standards:
1. ISO 9000
2. ISO 9001

Q.3 What is Quality planning?

Ans: Quality planning is selecting applicable procedures and standards for a particular project and modify these as required.
- A quality plan sets out the desired product qualities and how these are assessed and define the most significant quality attributes
- It should define the quality assessment process
- It should set out which organisational standards should be applied and, if necessary, define new standards

Quality plan structure:
- Product introduction
- Product plans
- Process descriptions
- Quality goals
- Risks and risk management
- Quality plans should be short, succinct documents

Q.4 What is Quality control?

Ans: Quality control is ensuring that procedures and standards are followed by the software development team.

Quality Control :
- Involves a series of inspections, reviews, and tests used throughout the software process

- Ensures that each work product meets the requirements placed on it
- Includes a feedback loop to the process that created the work product
- Combines measurement and feedback in order to adjust the process when product specifications are not met
- Requires all work products to have defined, measurable specifications to which practitioners may compare to the output of each process

Q.5 What do you mean by Software measurement and metrics?

Ans: Software measurement:

Software measurement is concerned with deriving a numeric value for an attribute of a software product or process. This allows for objective comparisons between techniques and processes

Metrics:

- Any type of measurement which relates to a software system, process or related documentation
- Allow the software and the software process to be quantified
- Measures of the software process or product
- May be used to predict product attributes or to control the software process

Process Improvement and Measurement

Q.1 State Process classification.

Ans: Processes can be classified as:

Informal: No detailed process model. Development team chose their own way of working

Managed: Defined process model which drives the development process

Methodical: Processes supported by some development method

Supported: Processes supported by automated CASE tools

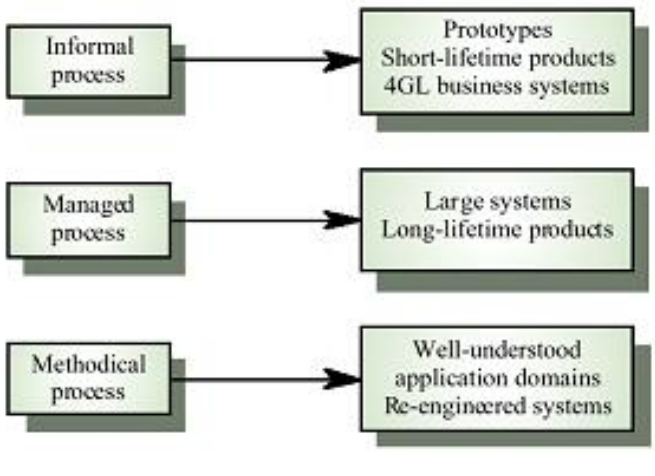


Figure 13.1 Process applicability

Q.2 What is Process analysis?

Ans: The study of existing processes to understand the relationships between parts of the process and to compare them with other processes.

Process analysis techniques

1)Published process models and process standards

-It is always best to start process analysis with an existing model.

-People then may extend and change this.

2)Questionnaires and interviews

-Must be carefully designed.

-Participants may tell you what they think you want to hear

3)Ethnographic analysis

-Involves assimilating process knowledge by observation

Q.3 What is Process modeling?

Ans: The documentation of a process which records the tasks, the roles and the entities used. Process models may be presented from different perspectives

Q.4 What is Process change?

Ans: It is one of the stages in process improvement. Here, changes to the process that have been identified during analysis are introduced.

Q.5 Explain CMMI process improvement framework.

Ans: The Capability Maturity Model Integration (CMMI) is a capability maturity model developed by the Software Engineering Institute of Pittsburgh, USA. The CMMI principal is that "the quality of a system or product is highly influenced by the process used to develop and maintain it". CMMI can be used to guide process improvement across a project, a division, or an entire organization.

CMMI provides:

- Guidelines for processes improvement
- An integrated approach to process improvement
- Embedding process improvements into a state of business as usual
- A phased approach to introducing improvements

CMMI Models consists of three overlapping disciplines providing specific focus into the Development, Acquisition and Service Management domains respectively:

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

CMMI Model Framework

Depending on the CMMI areas of interest (acquisition, services, development) used, the process areas it contains will vary. Process areas are the areas that will be covered by the organization's processes. The table below lists the process areas that are present, also called "core," to all CMMI areas of interest in CMMI Version 1.3. This collection of sixteen process areas is called the CMMI core process areas.

Capability Maturity Model Integration (CMMI) Core Process Areas

Abbreviation	Name	Area	Maturity Level
CAR	Causal Analysis and Resolution	Support	5
CM	Configuration Management	Support	2
DAR	Decision Analysis and Resolution	Support	3
IPM	Integrated Project Management	Project Management	3
MA	Measurement and Analysis	Support	2
OPD	Organizational Process Definition	Process Management	3
OPF	Organizational Process Focus	Process Management	3
OPM	Organizational Performance Management	Process Management	5
OPP	Organizational Process Performance	Process Management	4

Capability Maturity Model Integration (CMMI) Core Process Areas			
Abbreviation	**Name**	**Area**	**Maturity Level**
OT	Organizational Training	Process Management	3
PMC	Project Monitoring and Control	Project Management	2
PP	Project Planning	Project Management	2
PPQA	Process and Product Quality Assurance	Support	2
QPM	Quantitative Project Management	Project Management	4
REQM	Requirements Management	Project Management	2
RSKM	Risk Management	Project Management	3

Multiple Choice Questions

Chapter 1- Multiple Choice Questions

Q1. A software is :
- (a) A set of Instructions
- (b) A set of functions
- (c) A set of workspace
- (d) None of the above

Q2. A software doesn't-
- (a) Face out
- (b) Wear out
- (c) Clean out
- (d) None of the above

Q3. Sixty to Eighty percent of effort goes to:
- (a) Planning Phase
- (b) Coding Phase
- (c) Maintenance Phase
- (d) None of the above

Q4. System that analyzes the real world problems and meets strictly the timelines in execution are:
- (a) Open Systems
- (b) Real Time Systems
- (c) Hard systems
- (d) None of the above

Q5. SE is the establishment & use of sound engineering principles in order to obtain an:
- (a) Economic software
- (b) Reliable software
- (c) Efficient software
- (d) All of the above

Key:
1. (a)
2. (b)
3. (c)

4. (b)
5. (d)

.

Chapter 2-Multiple Choice Questions

Q1. Simplest oldest process model:
(a) RAD Model
(b) Waterfall Model
(c) DOD Model
(d) None of the above

Q2. A model in which a basic model is developed before the actual model-
(a) Prototype Model
(b) Waterfall Model
(c) DOD Model
(d) None of the above

Q3. RAD is:
(a) Random Application Development
(b) Rigid Application Development
(c) Rapid Application Development
(d) None of the above

Q4. Example of Evolutionary process model is:
(a) WINWIN Spiral Model
(b) Waterfall Model
(c) Prototype Model
(d) None of the above

Q5. Spiral Model was introduced by:
(a) Peckner
(b) Boehm
(c) Albert
(d) None of the above

Key:
1. (b)
2. (a)
3. (c)
4. (a)

5. (b)

.

Chapter 3- Multiple Choice Questions

Q1. Reviewing datelines and use of the project completion schedules is:
(a) Project Design
(b) Project Planning
(c) Project Maintenance
(d) None of the above

Q2. Project Planning can be done through:
(a) Prototype Model
(b) Waterfall Model
(c) Gantt Chart
(d) None of the above

Q3. PERT is:
(a) Parallel Evaluation and Review Technique
(b) Program Evaluation and Review Technique
(c) Program Evaluation and Ready Technique
(d) None of the above

Q4. CPM is:
(a) Critical Pan Method
(b) Critical Path Mode
(c) Critical Path Method
(d) None of the above

Q5. Project Scheduling could be done through:
(a) Project Design
(b) PERT
(c) Gantt Chart
(d) None of the above

Key:
1. (b)
2. (c)
3. (b)
4. (c)
5. (b)

Chapter 4- Multiple Choice Questions

Q1. Product capabilities are defined by:
(a) Functional Requirements
(b) Functional Planning
(c) Project Maintenance
(d) None of the above

Q2. Requirements that specify the constraints and specifies systematic approach to maintain quality:
(a) Functional Requirements
(b) Non Functional Requirements
(c) User Requirements
(d) None of the above

Q3. Interface specification includes:
(a) Description of requirements of the system
(b) Description of the interfaces of the system
(c) Both (a) & (b)
(d) None of the above

Q4. SRS includes the specification of:
(a) User requirement
(b) System requirement
(c) Both (a) & (b)
(d) None of the above

Key:
1. (a)
2. (b)
3. (b)
4. (c)

Chapter 5- Multiple Choice Questions

Q1. Process of Requirements elicitation; Requirements analysis; Requirements validation; Requirements management are included in:
(a) User Requirement
(b) Requirement Engineering
(c) Both (a) & (b)

(d) None of the above

Q2. SRS is:
(a) Software Requirement Source
(b) Software Requirement Specification
(c) System Requirement Source
(d) None of the above

Q3. Ways of structuring the requirements to represent the perspectives of different stakeholders are:
(a) Gantt Chart
(b) PERT
(c) View Points
(d) None of the above

Key:
1. (b)
2. (b)
3. (c)

Chapter 6- Multiple Choice Questions

Q1. To identify protection requirements that ensure that system failures do not cause injury or death or environmental damage is:
(a) Safety Specification
(b) Security Specification
(c) None of the above

Q2. The possibility that the defined goals are not met is:
(a) Fear
(b) Risk
(c) Remedy
(d) None of the above

Q3. The likelihood and consequences of the risks are assessed in:
(a) Planning
(b) Coding analysis
(c) Risk analysis
(d) None of the above

Q4. Risk Projection is also known as:
(a) Risk Identification
(b) Risk Estimation
(c) Risk Specification
(d) None of the above

Q5. Risk assessment is done during:
(a) Planning Phase
(b) Design Phase
(c) Coding Phase
(d) None of the above

Key:
1. (a)
2. (b)
3. (c)
4. (b)
5. (a)

Chapter 7- Multiple Choice Questions

Q1. Provide the scale for quantifying qualities:
(a) Plan
(b) Metrics
(c) Risk
(d) None of the above

Q2. Metrics that are used to check the progress of project development:
(a) Process metrics
(b) General metrics
(c) Both (a) & (b)
(d) None of the above

Q3. LOC is an example of:
(a) Proper measure
(b) Indirect measure
(c) Direct measure
(d) None of the above

Q4. Complexity is an:

(a) Indirect measure
(b) Identified measure
(c) Unidentified measure
(d) None of the above

Q5. RMMM is:
(a) Risk Mitigation, Monitoring and Management
(b) Risk Management, Monitoring and Modeling
(c) Risk Management, Mitigating and Modeling
(d) None of the above

Key:
1. (b)
2. (a)
3. (c)
4. (a)
5. (a)

Chapter 8- Multiple Choice Questions

Q1. TPS is:
(a) Transport Processing system
(b) Transaction Processing system
(c) Transport Parallel system
(d) None of the above

Q2. Which of the following is not an analysis technique?
(a) Ethnography
(b) Interviewing
(c) Planning
(d) None of the above

Key:
1. (b)
2. (c)

Chapter 9- Multiple Choice Questions

Q1. Agile methods focus on:

(a) Design
(b) Code
(c) Code and design
(d) None of the above

Q2. An act of developing new system through the usage of previously available components or other systems is:
(a) Software planning
(b) Software use
(c) Software reuse
(d) None of the above

Q3. Collections of abstract and concrete classes that can be adapted and extended to create application systems are:
(a) Application systems
(b) Applications
(c) Application frameworks
(d) None of the above

Key:
1. (b)
2. (c)
3. (c)

Chapter 10- Multiple Choice Questions

Q1. Provide the scale for quantifying qualities:
(a) Plan
(b) Metrics
(c) Risk
(d) None of the above

Q2. Metrics that are used to check the progress of project development:
(a) Process metrics
(b) General metrics
(c) Both (a) & (b)
(d) None of the above

Q3. LOC is an example of:
(a) Proper measure

(b) Indirect measure
(c) Direct measure
(d) None of the above

Q4. Complexity is an:
(a) Indirect measure
(b) Identified measure
(c) Unidentified measure
(d) None of the above

Q5. RMMM is:
(a) Risk Mitigation, Monitoring and Management
(b) Risk Management, Monitoring and Modeling
(c) Risk Management, Mitigating and Modeling
(d) None of the above

Key:
1. (b)
2. (a)
3. (c)
4. (a)
5. (a)

Chapter 11- Multiple Choice Questions

Q1. COCOMO is:
(a) Constructive Code Model
(b) Constructive Cost Model
(c) Complexity Cost Model
(d) None of the above

Q2. An algorithmic cost estimation method that computes effort, in person-months, from an estimate of size, measured in lines of code:
(a) Function Points
(b) COCOMO
(c) Both (a) & (b)
(d) None of the above

Q3. An algorithmic cost estimation method in which you count features of the requirements and use these to compute an estimate of the system's size:

(a) Function Points
(b) COCOMO
(c) Both (a) & (b)
(d) None of the above

Q4. Effort prediction model's general form:
(a) Effort = Productivity * solution
(b) Effort = Problem * solution
(c) Effort = Productivity * size
(d) None of the above

Q5. Costar is:
(a) An example of COOCMO
(b) An example of Function Points
(c) An example of Object Points
(d) None of the above

Q6. Application composition model is a submodel in:
(a) COOCMO I
(b) COOCMO II
(c) Prototype Model
(d) None of the above

Q7. An example of theoretical multivariable dynamic model is:
(a) COOCMO I
(b) COOCMO II
(c) PUTNAM
(d) None of the above

Key:
1. (b)
2. (b)
3. (a)
4. (c)
5. (a)
6. (c)
7. (d)

Chapter 12- Multiple Choice Questions

Q1. Establishing organizational procedures and standards for quality is:
(a) Testing
(b) Quality assurance
(c) Planning
(d) None of the above

Q2. The plan which sets out and defines the desired product qualities and the way they are assessed:
(a) Test Plan
(b) SDS
(c) Quality Plan
(d) None of the above

Q3. Quality Control involves a series of:
(a) Inspections & reviews
(b) Tests
(c) Both (a) & (b)
(d) None of the above

Q4. Software measurement is concerned with deriving a numeric value for an attribute of a:
(a) Software product
(b) Product effort
(c) Software process
(d) Both (a) & (c)

Q5. LOC is:
(a) Lines of Code
(b) A metric
(c) A code
(d) Both (a) & (b)

Key:
1. (b)
2. (c)
3. (c)
4. (d)
5. (d)

Chapter 13- Multiple Choice Questions

Q1. CMMI is:
(a) Capability Mathematic Model Integration
(b) Capability Maturity Model Integer
(c) Capability Maturity Model Integration
(d) None of the above

Q2. The study of existing processes to understand the relationships between parts of the process and to compare them with other processes is:
(a) Test Planning
(b) Process analysis
(c) Process Planning
(d) None of the above

Q3. Process analysis techniques:
(a) Inspections & reviews
(b) Questionnaires & interviews
(c) Both (a) & (b)
(d) None of the above

Key:
1. (c)
2. (b)
3. (b)

Case Studies

Case Study 1:

Consider a pen manufacturing industry, The HR department of the company wants to maintain the daily record of the employees of the manufacturing packaging department, accounts department and transportation department. HR department also wants the regular updation of the records. Suggest the best way to keep record of the data.

Case Study 2:

Consider again the case study 1, develop the model by using following model development paradigms:

(a) Waterfall
(b) RAD
(c) Prototype

Compare the difference in the development.

Case Study 3:

Consider the development of the admission management system of a college. If you are a given to make changes in the existing system, then which development model you would prefer to make updations.

Case Study 4:

If a person is appointed to develop software for a car manufacturing company to provide an automated system to keep control on the speed of the car and other security measures. Suggest the best development model for the developer to avoid the expected risks to be met by the car system. Develop a flow chart for the same. Also design SRS for the same.

Case Study 5:

Consider again the case 4, if the developer follows the RAD model for the development of the required system, then state the expected advantages or disadvantages.

Case Study 6:

Consider case study1, HR department want to present the datelines of the step by step development of the system. Suggest the best method to guide for its scheduling

Case Study 7:

In a bulb manufacturing industry, a new system is deployed to keep check on the quality of the product. Suggest the measures to verify the quality of the system as a tester.
Develop a test plan and check the system as per the designed plan.

References

- G. Singh, S. Puri, Software Engineering, Volume 1,
- Sommerville, Software Engineering, 6th Edition
- http://www.ecfc.u-net.com/cost/models.htm
- http://www.ccs.neu.edu
- http://www.vates.com/eng/cmmi5/what-is-cmmi.html
- http://www.dthomas.co.uk/dtalm/products/technologies/what-is-cmmi.htm
- http://www.rpl-blog.blogspot.com/2010/03/43-quality-planning.html
- http://www.cs.montana.edu

Keywords

Software, hardware, software development life cycle, failure curve, software myths, Management Myths, practitioners, Customer Myths, Practitioner's Myths, Software Engineering, System software, Real Time software, Business software, Engineering & Scientific software, Embedded software, Artificial Intelligence, Personal computer software, Web based software, expert system, knowledge-based systems, pattern recognition, Software Characteristics

Software process models, Common Process Framework, Task sets, project mile stones, work products, Waterfall Model, Linear Sequential model, Prototype Model, RAD Model, Evolutionary process model, Incremental Model, Spiral Model, WINWIN Spiral Model, DOD Model, Concurrent model, NASA Model, Component Based Development Model, Formal Methods Model, software Integration & Testing, Customer Evaluation, System Design Review (SDR), System Software Review (SSR), Software Requirement Specification (SRS), Preliminary Design Review (PDR), Critical Design Review (CDR), Software Design Specification (SDS), Test Readline Review (TRR), Functional Configuration Audit (FCA), Physical Configuration Audit (PCA)

Software project management, Proposal Writing, Project Planning & scheduling, Project Costing, Project monitoring & reviews, Personal Selection & Evaluation, Report Writing & Presentations, Quality Management, Configuration Management, four P's, Gantt charts, Work Breakdown Structure (WBS), activity network diagram, project plan, Program Evaluation and Review Technique (PERT), Critical Path Method (CPM), Risk Management, Functional requirements, Non-Functional Requirements, User requirements, System requirements

System requirements, Interface specification, software requirement document Requirements Engineering Processes, Feasibility studies, Requirements elicitation and analysis, Requirements validation, Requirements management System Models, Critical Systems Specification, Risk-driven specification, Safety specification, Security specification, Software reliability specification Software Metrics, software Measures, Process Metrics, Project metrics, Software Project Planning, Empirical, Putnam, COCOMO. Risk Identification and Projection: RMMM, Project Scheduling

and Tracking. Application Architectures, Data processing systems, Transaction processing systems, Event processing systems, Language processing systems, User Interface Design, Design issues, The user interface design process, User analysis, User interface prototyping, Interface evaluation, Rapid Software Development, Agile methods, Extreme programming, Rapid application development, Software prototyping. Software Reuse, Design patterns, Generator-based reuse, Application frameworks, Application system reuse, Software Evolution Verification, validation, Software inspections, automated static analysis, Verification and formal methods. Software Testing, System testing, Component testing, Test case design, Test automation, Software Cost Estimation, Software productivity, Estimation techniques, Algorithmic cost modeling, Project duration and staffing. Quality Management, Process quality, product quality, Quality assurance, quality standards, Quality planning, Quality control, Software measurement, metrics, Process Improvement, product quality improvement, Process classification, Process measurement, Process analysis, process modeling, Process change, CMMI, process improvement framework.