SOFTWARE ENGINEERING

UNIT-I

- SOFTWARE AND SOFTWARE ENGINEERING
  - THE NATURE OF SOFTWARE
  - THE UNIQUE NATURE OF WEB APPS
  - SOFTWARE ENGINEERING
  - SOFTWARE PROCESS
  - SOFTWARE ENGINEERING PRACTICE
  - SOFTWARE MYTHS
- PROCESS MODELS
  - A GENERIC PROCESS MODEL
  - PROCESS ASSESSMENT AND IMPROVEMENT
  - PRESCRIPTIVE PROCESS MODELS
  - SPECIALIZED PROCESS MODELS
  - THE UNIFIED PROCESS MODEL
  - PERSONAL AND TEAM PROCESS MODELS
  - PROCESS TERMINOLOGY
  - PRODUCT AND PROCESS

• THE NATURE OF SOFTWARE

Software is a collection of computer programs that when executed together with data provide desired outcomes.

IEEE defines software as: "Software is a collection of computer programs, together with data, procedure, rules, and associated documentation, which operate in a specified environment with certain constraints (conditions) to provide the desired outcomes". The view of computer software is shown in figure.

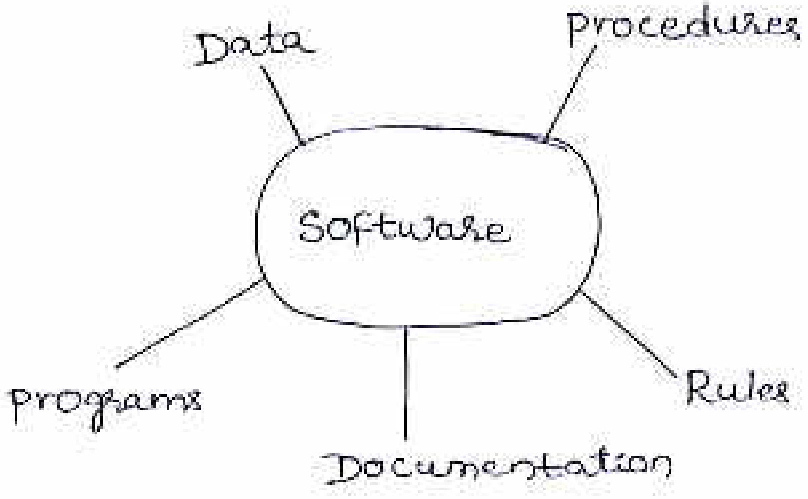


fig: Software View

Program : A program can be simple input – process – output statements, a function, a component, (or) program libraries. Software is developed by software engineers for an organization as per the requirements of a customer and it is used by the end users.

Data : Data are a collection of facts, measurements.

Procedures : procedures allow process steps.

Rules : Software rules are the guidelines for development, maintenance.

Documentation: Documentation is important in understanding the software code, design, constraints (conditions), customer needs, and specification for further maintenance.

The nature of software is used in different fields. Basically there are seven broad categories these are,

1. System software
2. Application software
3. Engineering/scientific software
4. Embedded software
5. product-line software
6. Web applications
7. Artificial intelligence software.

1. System Software:

System software is a collection of programs written to service other programs. The systems software is characterized by

→ Heavy interaction with computer hardware

→ Heavy usage by multiple users.

Ex: Operating systems, device drivers, compilers, editors.

2. Application Softwares:

Application software consists of standalone (independent) programs that solve a specific business need.

→ It facilitates business operations.

Ex: Educational softwares (Library management system), sale transaction softwares, video editing softwares, word processing softwares, spreadsheet softwares ~~etc~~ playing Audio/video files etc.

3. Engineering/scientific software:

Engineering problems and quantitative analysis are carried out using automated tools.
Scientific software is typically used to solve mathematical functions and calculations.

Ex: math calculation softwares, modelling and simulation softwares, Weather Report software, Engineering and scientific softwares, Automobile softwares such as CAD/CAM (computer-aided design/computer aided manufacturing softwares).

4. Embedded softwares:

Embedded software is a type of software that is built into hardware systems.

→ Embedded software is used to control, monitor and system functions.

Ex:→ Digital functions in Automobiles, dashboards display in railway stations, breaking systems, Washing machi

5. product-line software:

product-line software is a set of software intensive systems that share a common, managed set of features to satisfy the specific needs of market segments, Research and projects.

Ex: Database softwares, word processing softwares, multimedia softwares.

6. Web applications:

Web softwares has evolved from a simple website to search engines to web computing. Web applications are spread over a network such as internet.

Ex: Web softwares can be used in academic, business, Information management, which are developed by HTML, PHP, JSP.net etc.

7. Artificial Intelligence Softwares:

Artificial intelligence software is made to think like human beings and, therefore, it is useful in solving complex problems automatically.

Artificial intelligence software uses techniques or algorithms for writting programs to represent and manipulate knowledge.

Ex: Game playing, Robotics, speech recognition, medical diagnosis etc.

• THE UNIQUE NATURE OF WEB APPS:

The modern living is virtually depending on the world wide web also called Web. Web based systems are now available from stand-alone (independent) tools to corporate databases and business applications.

Web applications are also known as WebApps. Web apps deliver a wide range of facilities to the enduser. These are supported by various technologies such as HTML, ASP.net, JSP/Java, php, perl. There are various categories of web applications.

1. Web portals
2. Informational Systems
3. Interactive Systems
4. Transactional WebApps
5. Web Services
6. Online communities.
7. Social Web Apps
8. Shared data
9. Ubiquitous.

1. Web portals

A Web portal is a customized Website that provides information from various sources in a consistent manner.

Ex: Goverment portals, Business portals, Online shopping mall etc.

2. Informational systems:

These systems provides content wise information with navigation and links.

Ex: Online newspapers, manuals, online books, online classifieds etc.

3. Interactive Systems:

These provide an interface to the users to interact with the system functionalities.

Ex: Bulletin boards, Registration forms, Online games etc.

4. Transactional WebApps:

Such systems provides functionalities for business Operations.

Ex: Online banking, online reservation etc.

5. Web Services:

Web Services is a standardized way to propagate communication between the client and server applications on the Web.

Ex: online booking, enterprise applications, Business Apps.

6. Online communities:

These are the groups that share information.

Ex: Electronic shopping malls, online market places, discussion groups etc.

7. Social WebApps:

These WebApps help people to link each Other.

EX: Social networking Websites (facebook, twitter) etc.

8. Shared data:

These are the applications where shared data repositories (storages) managed and used by the practitioners (Developpers (or) programmers).

EX: Data warehouses, Online repositories.

9. Ubiquitous:

These applications allow accessing web based contents and services through different devices. Such as,

EX: computers, personal Digital Assistants, mobile phones, set top boxes connected to TV's etc.

EX: customized services.

* characteristics of WebApps:

The nature of WebApps development is multidisciplinary it handles information in various formats such as text, graphics, video, audio, etc. Key characteristics of WebApps.

1. User satisfaction
2. performance
3. security
4. Usability
5. Reliability
6. Availability
7. Concurrency
8. Feedback.

1. User satisfaction:

WebApps must provide all the expected functionalities to the user without much effort.

2. performance:

WebApps must respond quickly & fast to the user. User doesn't want to wait for a long time to execute the next Operation.

3. Security:

It is the capability of a system to allow minimum chance of malicious or accidental attacks and prevent disclosure or loss of information.

4. Usability:

It specifies how easy application is to use for various category of users.

5. Reliability:

It is the ability of a system to continue operating without failure over a given time intervals.

6. Availability:

Web services must provide 24X7 facilities to the user for Operation and maintenance.

7. concurrency:

WebApps are accessed concurrently by many Users at the same time.

8. Feedback:

It gives freedom to the users to share their experience of using the WebApps. This helps to improve the applications.

* Web Apps Development process :

Web based development is the area of web engineering. web applications deliver a complex array of content and functionality to a wide variety of end-Users.

The figure shows the main phases of Web Apps development process are.

1. Requirements
2. modelling
3. prototype & construction
4. Testing & quality Assurance
5. Deployment
6. Maintenance.

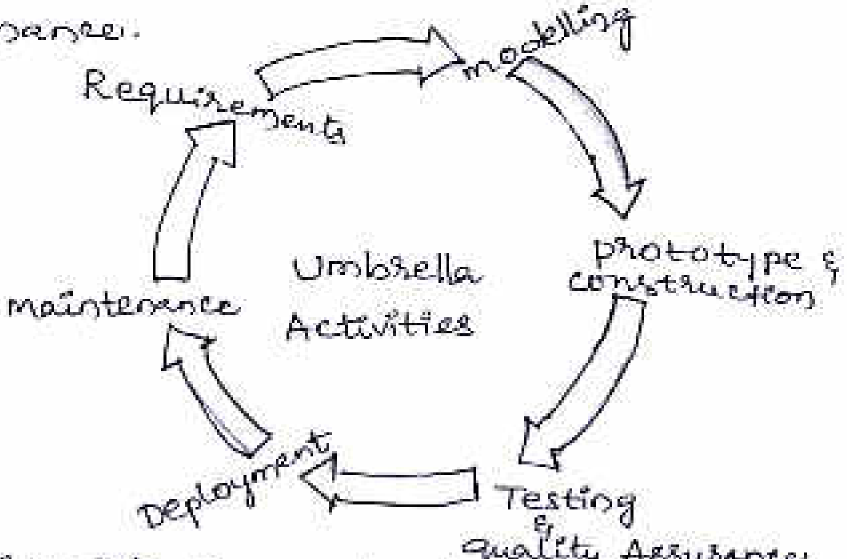


fig: Web Apps development process.

1. Requirements: User requirements are gathered and analyzed.

2. modelling: creating an architectures, it helps to navigation design, controls and user interface design.

3. prototype & ~~quality assurance~~ construction

This is the coding phase. prototype is the best way to develop an effective and good WebApp.

4. Testing & quality Assurance: Various tests are conducted to remove the presence of errors & quality is maintained.

5. Deployment: User manuals are prepared and provided to the end User.

6. Maintenance: maintenance is the ongoing activity after a WebApp is developed and deployed.

Umbrella activity: It support activities performed for an effective development of web Apps.

• SOFTWARE ENGINEERING

Software engineering definition:

Software engineering is a technological, managerial, discipline, concern with systematically develop the entais software product.

IEEE defines software engineering as:
"The systematic approach to the development, operation, maintenance and retirement of software".

The main goal of software engineering is to understand customer needs and develop software with improved quality, on time, and within budget.
The view of software engineering is shown in figure.
the productivity of programmers can be improved systematic approach is used in production and maintenance of software.

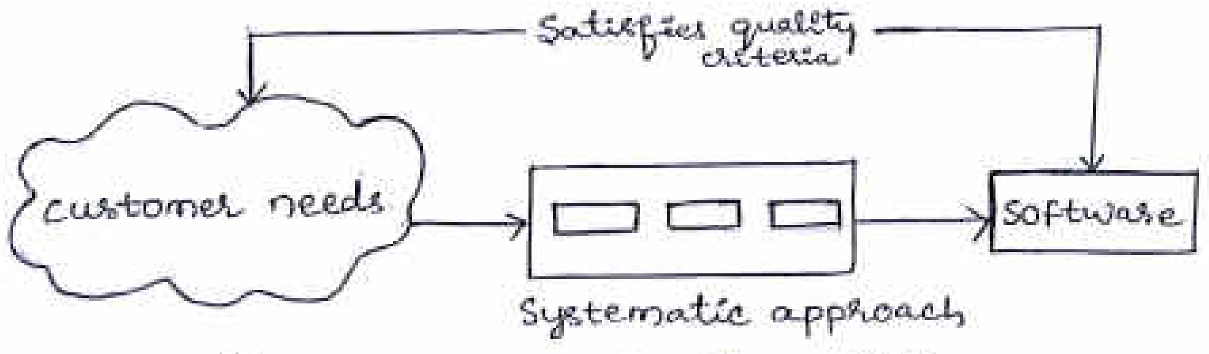


fig: software engineering view.

The main focus of software companies is to satisfy customer needs.

The software engineering approaches focus on providing disciplined practices for software development and maintenance that can satisfy customer and software engineers.

The Industries are also focussing on increasing productivity, improving the process, reduce rework, decreasing cycle time.

SOFTWARE PROCESS :

The concept of "software process" was introduced between 1950s and 1960s. At the time, the intention of software practioners was mainly to develop software using traditional programming languages, such as Assembly language, FORTRAN, COBOL, and so on.
The developers were using ad hoc processes for software development due to alternative technologies and lack of expertise and resources.
with time, the needs of people have grown and technology has become advanced.
In this way, it was urgent for software practitioners to introduce the systematic software process to produce quality software products.

The figure shows process, project and product.

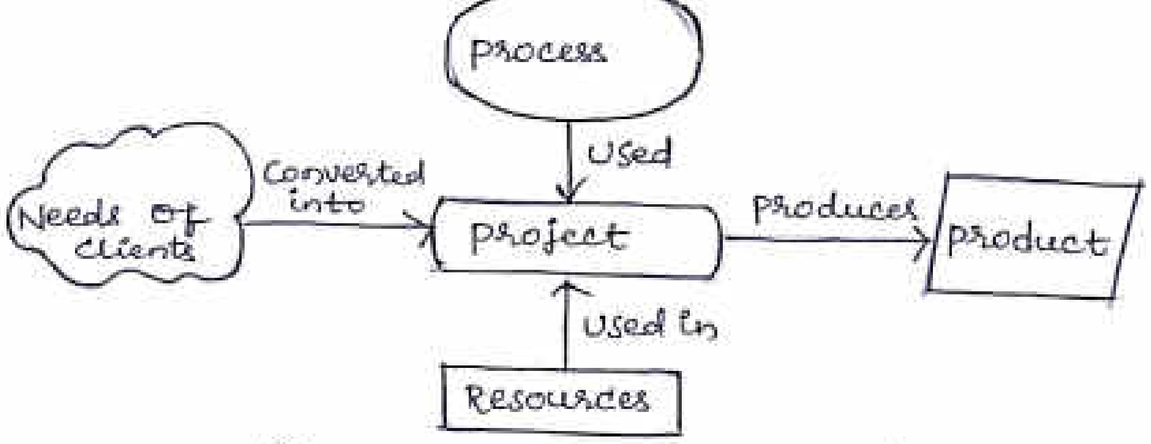


fig: process, project and product.

→ A software process is a set of ordered activities carried out to produce a software product. Each activity has well defined objective, task, and outcome.

→ A software project is an entity with defined start and end, in which a software process is being used. Software project is a cross functional entity which is developed through a series of projects using the required resources and project constraints (cost, schedule, quality).

→ A product: is the outcome of a software projects produced through process.

(ii) Software engineering principles:

The principle of software engineering are the general considerations that the software developer and the development organizations must follow while producing the software. These principles deals with the various software development activities in order to achieve the quality goals. Some important principles are

1. Focus on customers' problems, needs, priorities and expectations.
2. Choose appropriate process model.
3. Decomposition and modularity
4. Abstraction
5. Encapsulation
6. Incremental development.
7. Understandability
8. consistency and completeness
9. Generality
10. perform verification and validation to maintain quality.
11. Follow-up the scope statements, deadlines and early product delivery.
12. Design for change
13. Follow disciplined and mature process
14. Take responsibility and commitment.
15. Better planning and management rather than Technology.

1. problem understanding:

TO understand the exact problem and requirements that the customer has is a difficult task in the overall software development and maintenance process. There are several issues involved in problem understanding. Usually, customers are from different backgrounds and they do not have a clear understanding of their problems and requirements. Also, the customers don't have technical knowledge.

The lack of communication among software engineers and customers causes problems for the software engineers in clearly understanding the customer needs.

2. Quality and productivity:

Software productivity and quality have become the most considerable challenges in the development of software. Software engineering practices mainly emphasize providing quality product in a small cycle time. ~~Goods~~ Quality products provide customer satisfaction.

A good quality products implements features that are required by the customer. The quality attributes are Reliability, usability, efficiency, performance, maintainability, portability, functionality.

3. cycle time and cost:

A systematic software engineering approach can reduce the cycle time and product cost.

4. Reliability:

Reliability is one of the most important quality attribute. Reliability is the successful operation of software within the specified environment and duration under certain conditions.

5. Change and maintenance:

change and maintenance in software come when the software is delivered and deployed at the customer site. They occur if there is any change in business operation, errors in the software, or addition of some new features. But change and maintenance are not easy tasks.

Change and maintenance in software are flexible but they are a very expensive task. sometimes, the maintenance cost and rework costs becomes more than the development costs.

6. Usability and Reusability:

Usability means the ease of use of a product in terms of efficiency, effectiveness, and customer satisfaction.

Reusability means the existing software components their development has become an institutional business in the modern software business scenario.

7. Repeatability and process maturity:

A software engineering process can be repeated in similar projects, which improves productivity and quality.

A mature software process produces quality products and improves software productivity. There are several standards such as ISO, CMM and Six Sigma, which emphasize process maturity and its guidelines.

8. Estimation and planning:

present estimation methods, such as lines of code (LOC), functional points (FP), are sometimes unable to accurately estimate project efforts. It is observed that the projects failure ratio is greater than the success rates. most of the projects fail due to underestimation of budget and schedule. planning technique is the important challenge for the practitioner.

Scanned by CamScanner

(ii) Software engineering principles:

The principle of software engineering are the general consideration that the software developer and the development organizations must follow while producing the software. These principles deals with the various software development activities in order to achieve the quality goals. Some important principles are

1. Focus on customers' problems, needs, priorities and expectations.
2. Choose appropriate process model.
3. Decomposition and modularity
4. Abstraction
5. Encapsulation
6. Incremental development.
7. Understandability
8. Consistency and completeness
9. Generality
10. Perform verification and validation to maintain quality.
11. Follow-up the scope statements, deadlines and early product delivery.
12. Design for change
13. Follow disciplined and mature process
14. Take responsibility and commitment.
15. Better planning and management rather than technology.

• SOFTWARE MYTHS :

Myths mean misleading attitude creates some serious problems.

(OR)

Myth mean false expectations (B) misinformation that creates problems. Myths are basically three categories.

1. Management myths
2. practioner myths
3. customer myths.

1. Management myths

managers are the main responsible stakeholders in development team. They are always in a situation to deliver the product within budget and schedule. In this connection or situations, there are following common management myths.

Myth ①: Adding more manpower can reduce the project schedule.

Myth ②: Management can outsource (Third party) the project to reduce the burden

Myth ③: Team members are known and adhere (follow) to all the information, procedures and standards.

Myth ④: Automated tools can solve the development problems.

2. practitioner myths:

people think that software development means only the programming part. But program development now has become an engineering discipline. Some common practitioner myths are as follows.

Myth ①: Once we write the program and get it to work, our job is done.

Myth ②: until I get the program running, I have no way of assessing its quality.

Myth ③: The success of software depends on the delivered product quality.

Myth ④: Documentation slows down the progress of the project.

3. customer Myth:

customers are the main stakeholders of the project. But they don't think from the managers and developers perspectives. The real facts about development are known to the developers and managers only.

This may lead to false expectations and dissatisfaction among the customers.

Myth ①: The general statements of the problem is sufficient to development.

Myth ②: Software is flexible to change and accomodate.

Myth ③: The customer think that, after delivery of the product ~~to~~ the ~~customer~~, to quality is check.

• PROCESS MODELS

• A GENERIC PROCESS MODEL :

A process model is a set of activities that have to be accomplished to achieve the process objectives. process models specify the activities, work products, relationships, milestones, etc. Some examples of process models are data flow model, life cycle model, quality model. A generic view of the software process model is shown in fig.

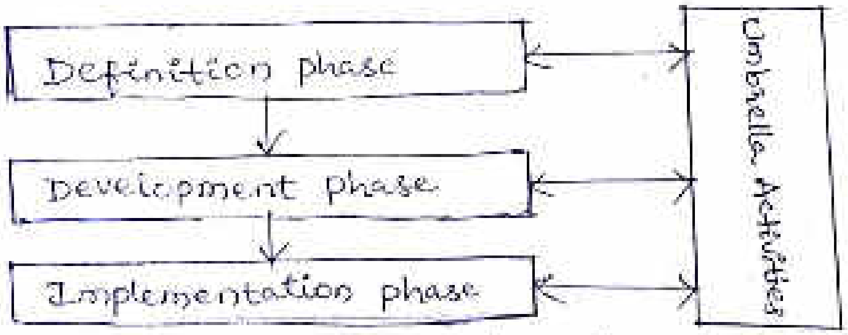


fig: A generic representation of process model.

The generic process model has three phases that are coordinated and supported by Umbrella activities. These are

1. Definition phase
2. Development phase
3. Implementation phase.

1. Definition phase:

This phase concentrates on understanding the problem and planning for the process model. The activities may include problem formulation, problem analysis, system engineering, and project planning for the process.

2. Development phase:

This phase focusses on determining the solution of the problem with the help of Umbrella activities. The main activities of this phase are desining the architecture and algorithms of the system, writing codes and testing the software.

3. Implementation phase:

Deployment, change management, defect removal, and maintenance activities are performed in this phase.

The Umbrella activities are responsible for ensuring the proper execution of definition, development and implementation phases. These phases are managed and controlled by the umbrella activities.

The umbrella activities are project management, quality assurance, configuration management, risk management, work product preparation and deployment and process improvement.

PROCESS ASSESSMENT AND IMPROVEMENT PROCESS:

• PROCESS ASSESSMENT AND IMPROVEMENT PROCESS:

→ process assessment is an activity in which it is ensured whether the software meets the basic criteria for "successful software project".

→ The ultimate goal of improvement in a process is to enable the organization to produce more quality products.

→ process improvement is an incremental improvement of process, which is used for software development. sometimes, it becomes very difficult to apply as improved software process to achieve the specified results due to short delivery span, insufficient knowledge of the process, insufficient managerial support. ~~and~~

→ There exist various improvement process models, such as CMMI (Capability maturity model Integration), Six Sigma.

→ All these process models provide improvement guidelines and standards for improving software process.

→ cmmI levels are Level 1: Initial, Level 2: Repeatable, level 3: Defined, level 4: managed, level 5: Optimizing.

→ Six Sigma frame work activities are planning, Highlevel design, Review, Development, postmortem.

→ ISO 9001:2000 quality standard.

. PRESCRIPTIVE PROCESS MODEL :

All process models having series of activities are known as "prescriptive process model". Each activity in the model is reffered to as a phase.

Generally activities include feasibility study, analysis, design, coding, testing, implementation, and maintenance. collectively these activities are called the "software development life cycle (SDLC).

The SDLC provides a framework that the activities performed to develop and maintain software.

various prescriptive process models proposed for software development, some of these models are waterfall model, prototyping model, incremental model, spiral model, Agile process model, RUP process models and so on.

Software development organizations follow some life cycle model for each project when developing a software product.

· Phased life cycle Activities (SDLC)

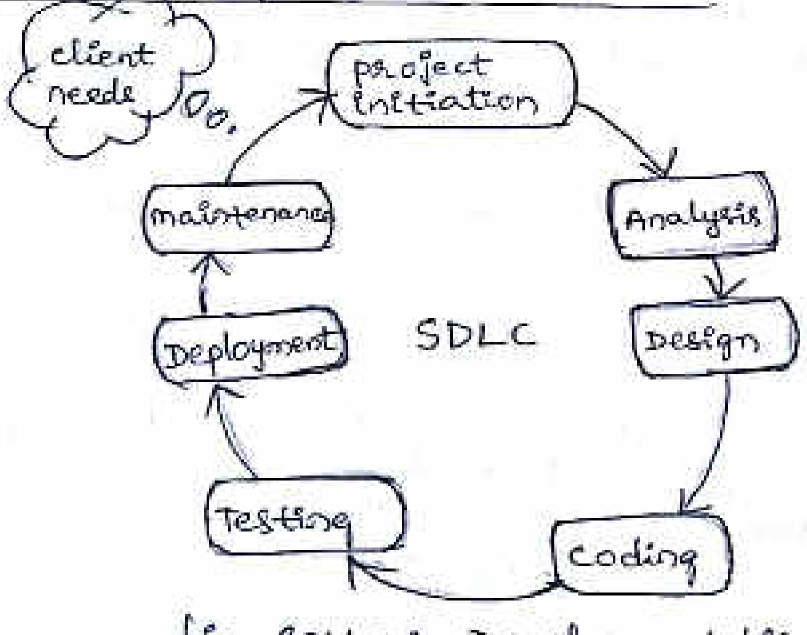


fig: Software Development Life cycle (SDLC).

* Project Initiation:

The aim of project initiation is to study the existing system; determine the feasibility of a new system, and define the scope, key elements, and a plan for the successful completion of the project.

Project initiation involves preliminary investigation, feasibility study, and project plan. Preliminary investigation is the initial step that gives a clear picture of what actually the physical system is.

Feasibility study performed such as technical feasibility, Economical feasibility, Operational feasibility.

Technical feasibility refers to Software & Hardware.

Economical feasibility refers to cost (or) budget, Operational feasibility refers to How to utilise (or) operate the system successfully.

* Requirements Analysis:

Requirement analysis is the process of collecting factual data, understanding the process involved, defining the problem. Analysis is Analysed by the ~~Analyser~~ Analyst.

The requirement analyses phase consists of three main activities

Requirement elicitation.

Requirement Specification.

Requirements verification and validation.

Requirement elicitation:

Requirements elicitation is about understanding the problem.

Requirement Specification(SRS):

SRS describes the user requirement, system requirement and functional requirements and non functional requirements with proper documents.

Requirements verification and validation:

Software design:

Software design focusses on the solution domain of the project on the basis of the requirement document prepared during the analysis phase.

The goal of the design phase is to transform the collected requirements into structure (or) Architecture. Architecture is suitable for implementation in programming languages.

The architecture is designed by the designer.

coding:

The coding phase is concerned with the development of the source code. This code is written in a formal languages called programming languages, such as

* Assembly language, C, C++, Java etc.

good code effots ~~to~~ can reduce testing and maintenance tasks. coding part is developed by the "practitioner" or developer (or) programmer.

Testing:

Testing is the process of executing the program with the intent of findout the errors or deffects, and removed.

Testing is performed at different levels. unit testing, integration testing, system testing and acceptance testing. Testing is tested by the tester.

Deployment (Release) (or) Transition:

After acceptance by the customer during the testing phase, deployment of the software ~~dee~~ begins. The purpose of software deployment is to make the software available for operational use. The deployers are the deploy the product along with user manuals.

Maintenance:

The maintenance phase comes after the software ~~pro~~ product is released and put into operation through the deployment process. Software maintenance is performed "adding new features".

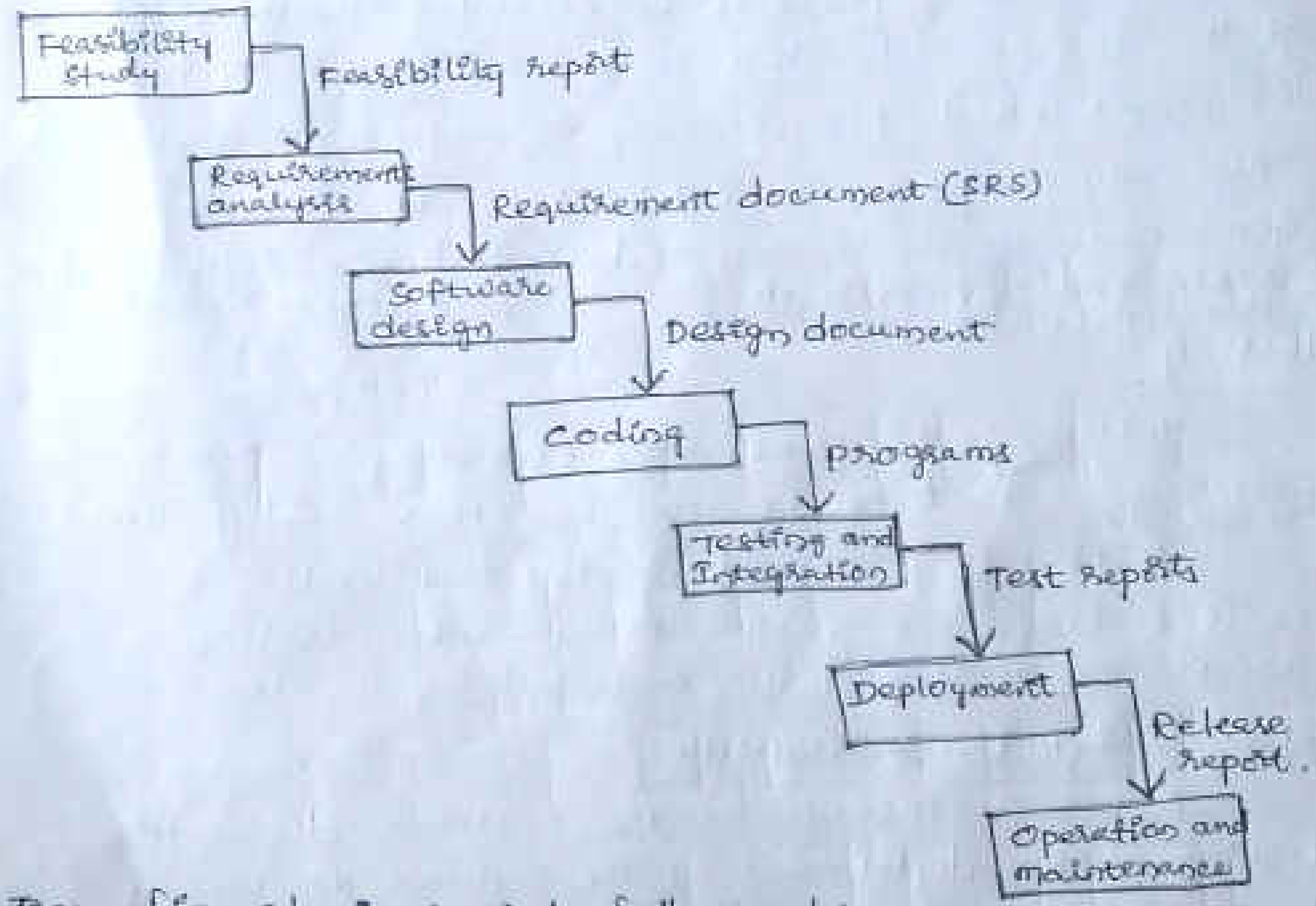
* Waterfall Model:

→ classical waterfall model
→ Iterative waterfall model

classical waterfall model:

The waterfall model is a classical development process model proposed by R. Waker Royce in 1970.

Software development proceeds through an orderly sequence of transitions from one phase to the next phase in order (like a waterfall). It is the simplest and the most widely used model in development. The classical waterfall model is shown in figure.



~~Den~~ fig: Classical waterfall model.

Development runs through a number of phases namely, feasibility study, analysis, design, coding, testing and integration, deployment, operation and maintenance.

This model development proceeds downward to the next phase. And there is no concept of backtracking in this model.

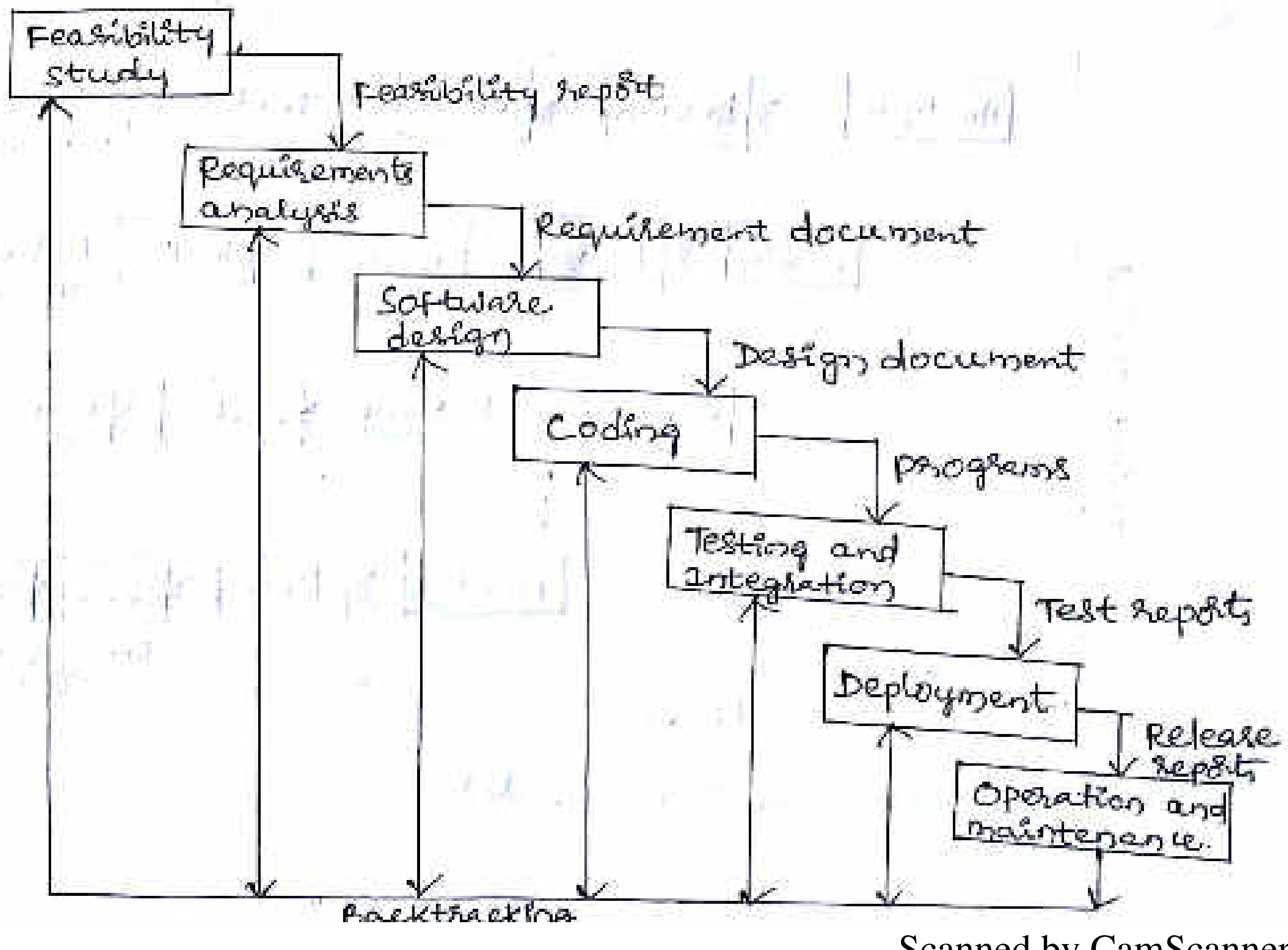
The waterfall model divides the set of phases. the model begins with feasibility study. The feasibility study prepare the project plan, describing the project scope, budget, schedule and software requirement specification (SRS). SRS is transformed into architectural design. A detailed design produces algorithms, which are further implemented in some programming languages. The source programs are tested to find the uncovered errors, after that the product is deployed at the customer.

Advantages: The main advantage of the waterfall model is that it is easy to understand and implement. Low cost.

Disadvantages: The disadvantage of the waterfall model is does not go back phase. overcome this model the iterative waterfall model is introduced.

Iterative Waterfall model:

fig: Iterative waterfall model.



The iterative waterfall model is an extended waterfall model with backtracking at each phase to its preceding phases. This was proposed by R.W. Royce in 1970.

In the iterative waterfall model, errors can be detected in almost every phase of development. If any error is introduced at any stage, there are two possibilities of its source of detection. Errors may be detected in the same phase in which they are introduced. The other possibility is they may be detected in the preceding phases of the phase in which they were introduced. for example, design errors may be detected at the analysis phase.

<u>Advantages</u>: The iterative waterfall model follows the Back-tracking process.

<u>Incremental Model:</u>

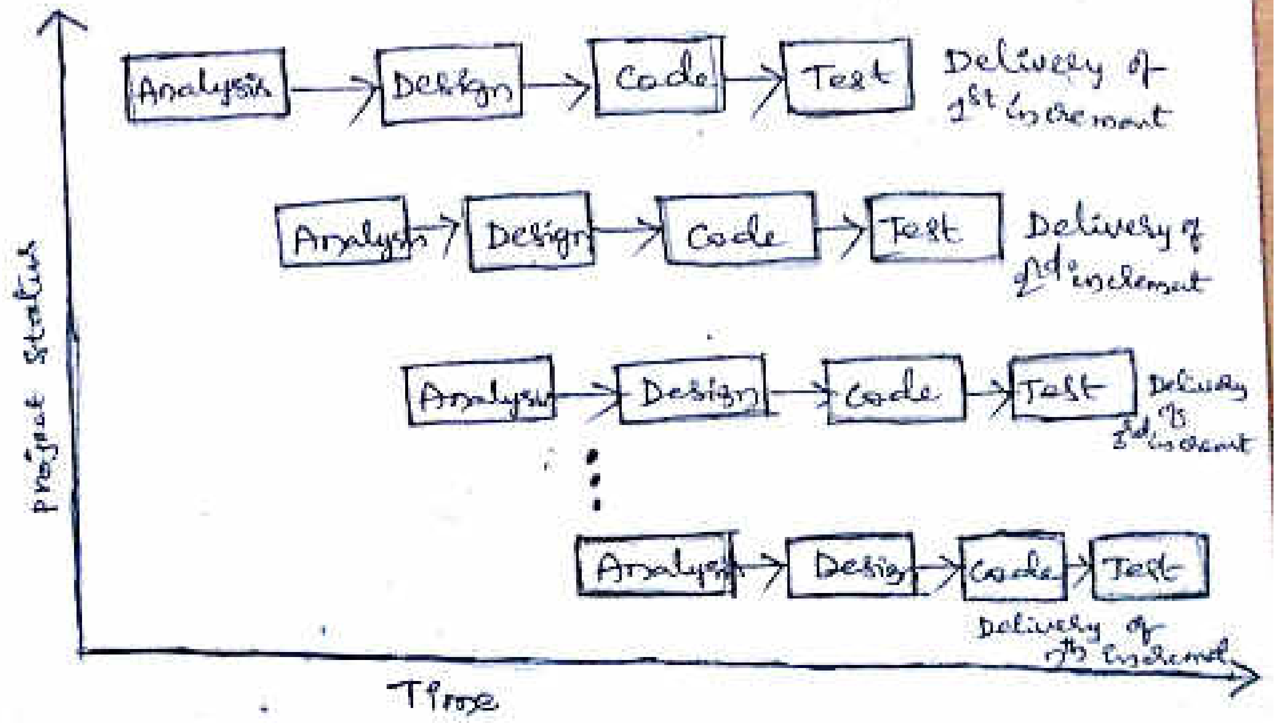


fig: Incremental model.

Prototyping Model:

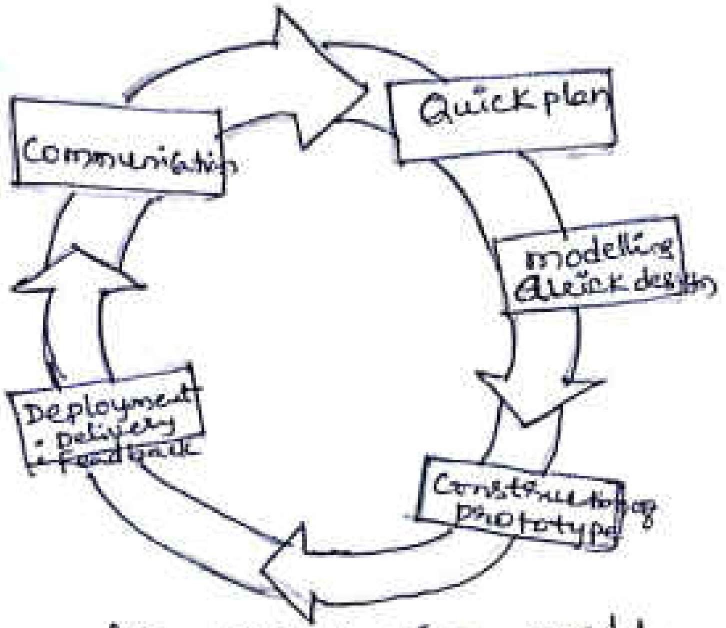


fig: prototyping model.

· prototyping pardigm begins with commenication. The Software engineer and customer meet and define the overall objectives for the software, identify whatever requirements are known.

prototyping iteration is planned quickly and modeling occur. The quick design leads to the construction of a prototype. The prototype is deployed and then evaluated by the customer.

Iteration occurs as the prototype is tuned to satisfy the needs of the customer, while at the same time enabling the developer to better understand what needs to be done.

Advantages:

The prototype serves as a mechanism for identifying software requirements.

The developer attempts to make use of existing programs.

The Spiral model:

The spiral model is originally proposed by Boehm in 1988. It is an evolutionary software process model, that is an iterative nature. The following figure shows spiral model

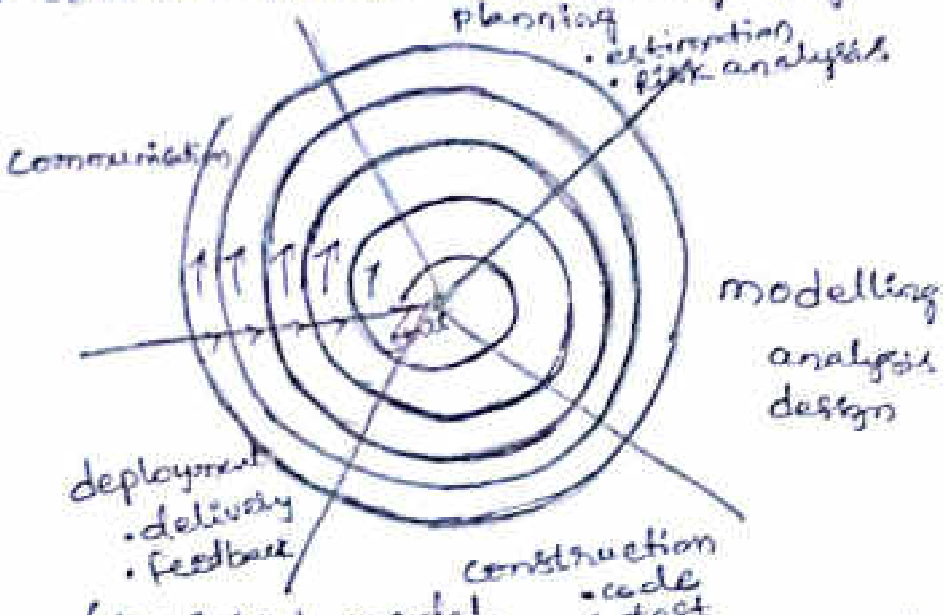


fig: Spiral model.

Using the spiral model, software is developed in a series of evolutionary releases. During early iterations.

In this model, activities are organized as a spiral with many loops. The exact number of loops in the spiral is not fixed.

The main focus of this model is identification and resolution of potential risks (product risks, project risks, process risks), with the help of risk analysis.

The project manager planned and focus on the number of iterations required to complete the software.

Advantages:

→ It handles high schedule projects and high budget projects.

→ It findout the early potential risks.

SPECIALIZED PROCESS MODELS:

The Specialized process models provides the quality software products. These are

- client-server software engineering
- Service-oriented software engineering
- Usability engineering process
- Model-driven software engineering.
- Aspect-oriented software engineering.

→ client-server software engineering:

In a client-server model, the client process makes use of the services provided by the server process.

The client-server model provides a convenient way to interconnect programs that are distributed efficiently across different locations. The example of client-server system are database servers, fileservers, transaction servers, groupware servers etc.

"client-server" describes the relationship between two computers programs. The client makes a service request from another program and the servers fulfils the request. the interaction of the client and server process follows a request-response pattern, is shown in figure.

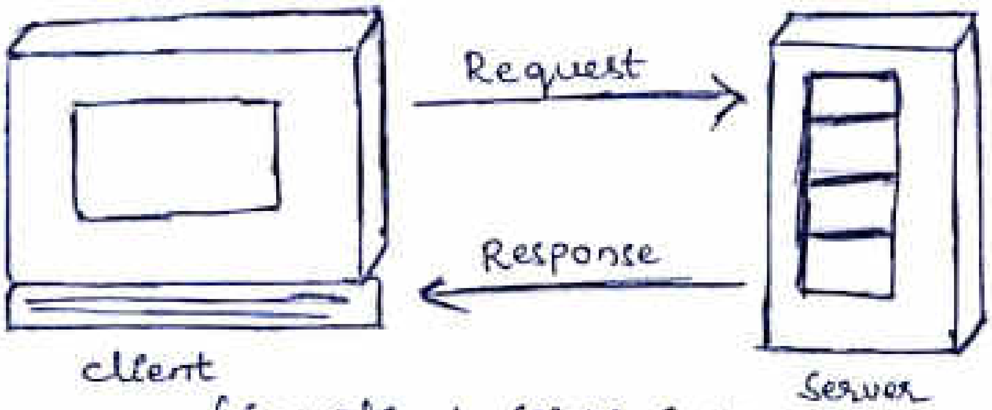


fig: client-server interaction.

The client server model consists of two major architecture types namely,

1. Two-tier architecture
2. Three-tier architecture.

Two-tier architecture:

Two-tier architecture is the simplest of the architecture types, consisting of only the server and the client application. It is shown in figure.

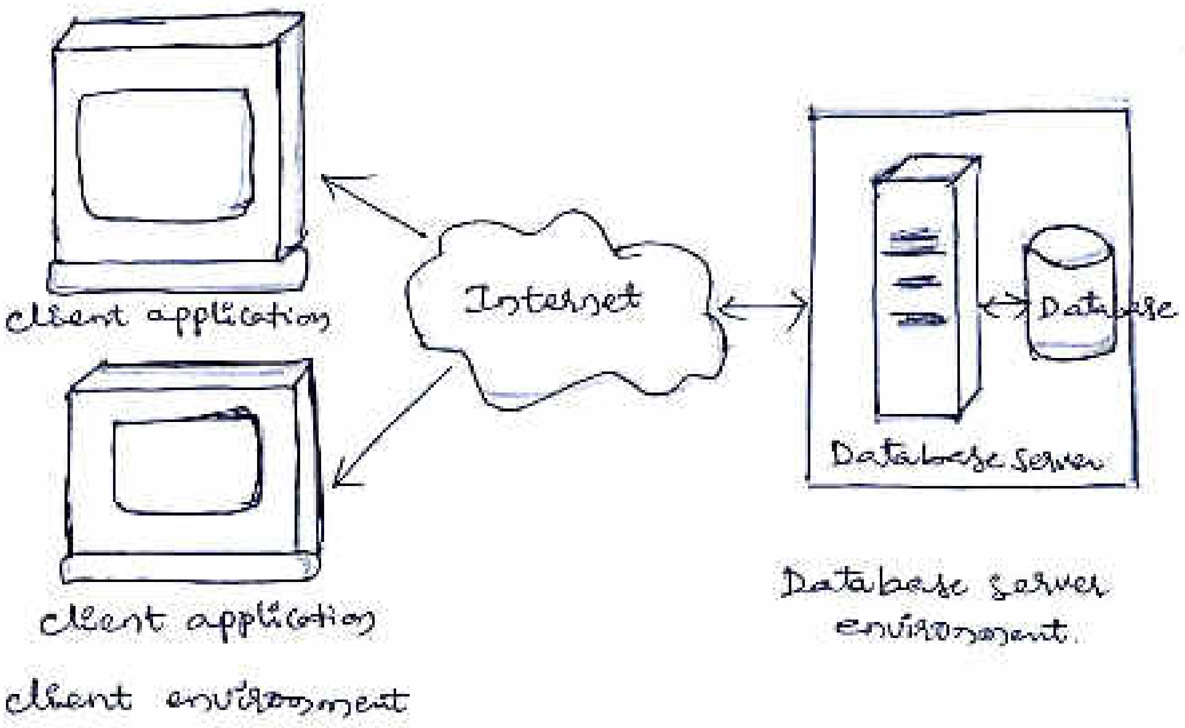


fig: Two-tier architecture.

The two-tier architecture, the server and the other user system interface is usually located in the clients environment and database management services are usually in a server, which is a more powerful machine that services many clients.

The two-tier architecture is a good solution for distributed computing interacting on a LAN simultaneously.

Three-tier architecture:

Three tier architecture is also known as "multi tier architecture", emerged to overcome the limitations of the two-tier architecture.

In the three-tier architecture, a middle tier was added between the user system interface client environment and database management server environment.

Three tier architecture has been shown to improve performance for groups with a large number of users and improves flexibility when compared to the ~~the~~ two tier approach.

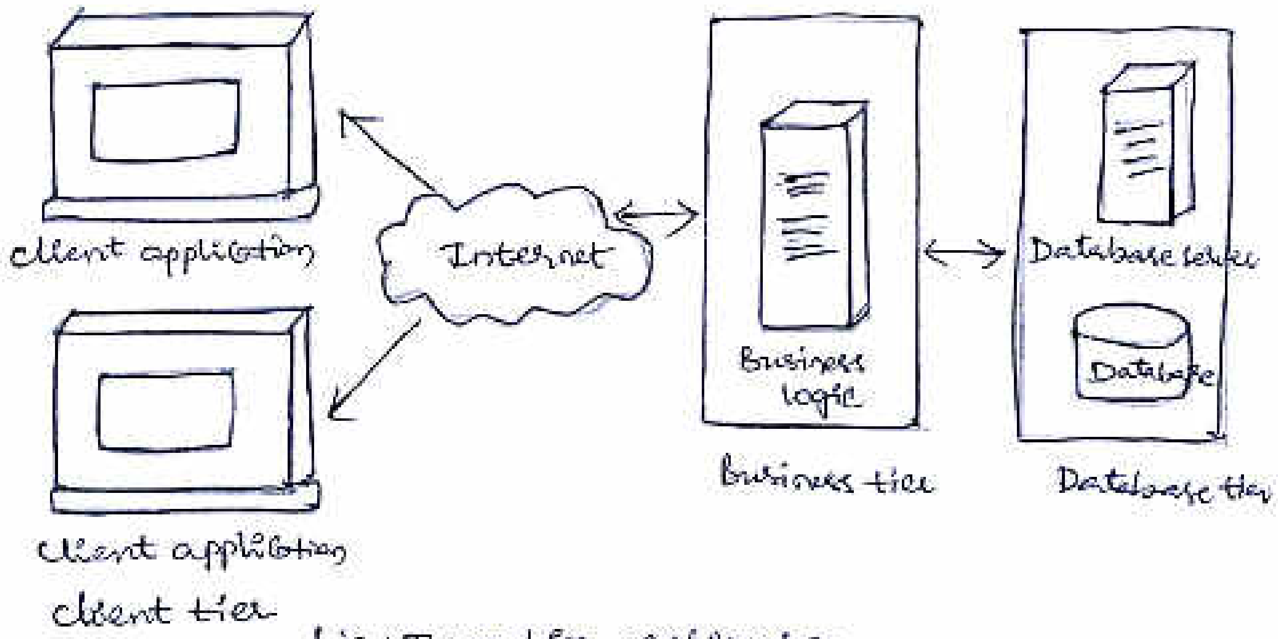


fig: Three tier architecture.

Applications:

client server applications are important in offering services throughout the internet. Some of the applications are

1. Mobile computing
2. Net banking
3. Online education system
4. Electronic commerce
5. Satallite communication
6. Military system
7. Remote procedure calls (RPC).

Service-Oriented Software Engineering:

The software market has grown rapidly in the last two decades, especially in web applications. As more software companies and organizations enter the web applications market, the number of web based capabilities and services available to software developers are rapidly increasing on the internet.

the Service Oriented Software engineering (SOSE) deals with theories, principles, methods, and tools for building enterprise scale solutions.

principles of Services in SOSE :

1. Loosely coupled
2. Abstract
3. Reusable
4. Composable
5. Stateless
6. Discoverable
7. service autonomy
8. service normalization

Usability Engineering;

Usability is one of the key features of software.

"Usability is the extent to which a product can be used by the specified users to achieve the specified goals with effectiveness.

Usability has the following five quality components.

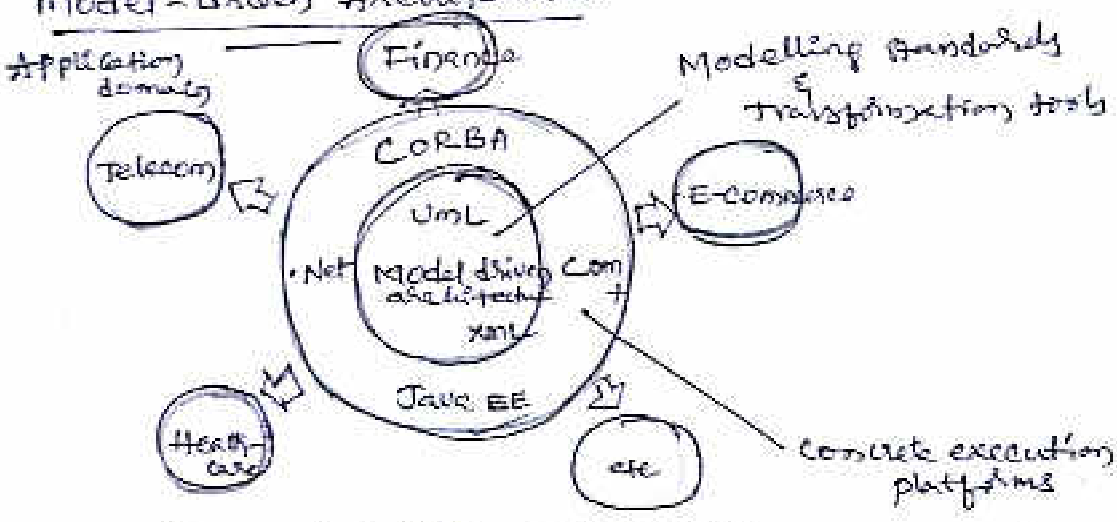
1. Ease of learning
2. Efficiency of use
3. Memorability
4. Error frequency and severity
5. Satisfaction

* Model-driven software Engineering (MDSE):

Model-driven software engineering is becoming a widely used approach for developing complex applications.
Some of applications are.

1. Finance
2. Telecom
3. Healthcare
4. e commerce.

model-driven Architecture:



Application domain
Finance
Modelling standards & transformation tools
Telecom
CORBA
UML
E-Commerce
.Net
Model driven architecture
Com +
XML
Java EE
Health care
etc
Concrete execution platforms

fig: model driven architecture.

model driven architecture is an approach to application design and implementation. These applications are depends on Unified modelling language (UML), CORBA, J2EE, .Net, XML ~~and~~ These are web Technologies platforms independently.

* Aspect-Oriented software engineering:

There are various programming approaches for developing good quality software products. The procedural approach has been widely used to develop software but the representation of real world entities and complexity management in software development were the major limitations.

The activities of AOSD

1. Aspect-Oriented Requirements Engineering (AORE)
2. Aspect-Oriented Architecture (AOA)
3. Aspect-Oriented design (AOD)
4. Aspect-Oriented programming (AOP)
5. Verification of aspect-oriented programs.

THE UNIFIED PROCESS MODEL:

The unified process is a frame work for Object Oriented models. This model is called as "Rational Unified process model (RUP)". It is proposed by Jacobson, Grady Booch.

This model is iterative and incremental nature.

There are four life cycle phases in unified process.

1. Inception
2. Elaboration
3. Construction
4. Transition.

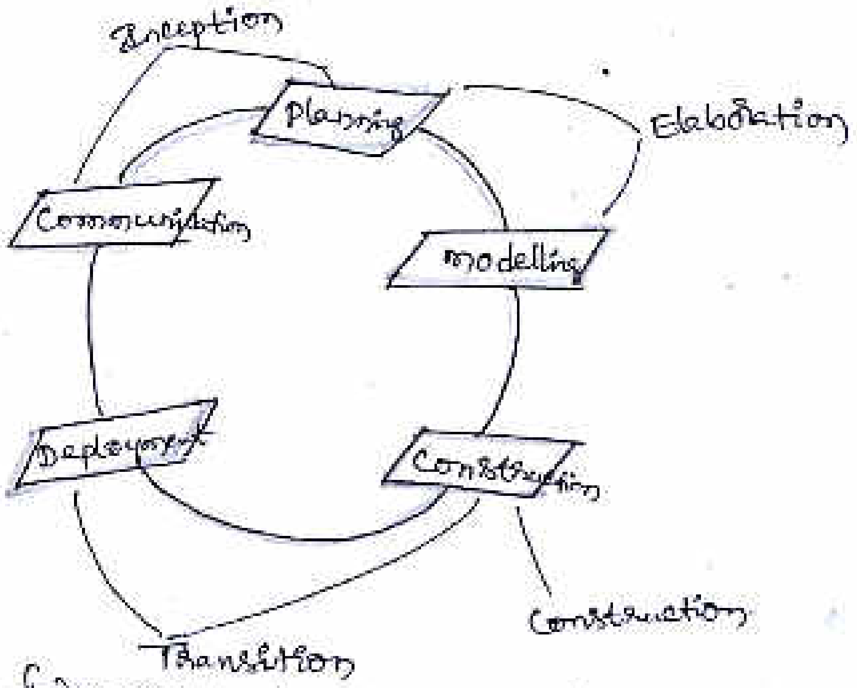


fig: Unified process model.

Inception phase:
In this phases There are two major activities that are conducted. communication and planning. It is mainly focus on project plan. usecase model and risk assessment.

Elaboration phase: In this phase there are two major activities that are conducted, planning and modelling. It mainly focus on Requirement analysis model, architectural model.

Construction phase: In this phase source code is developed and Test plan and test cases and User manuals, installation manual

Transition phase: It is delivery software products and User feedback.

PERSONAL AND TEAM PROCESS MODELS:

The quality of product can be better if it's designed around the people who work on the project.

The personal software process and team software process models are the frameworks, which are intended to improve the levels of quality in personal and team's software.

* personal software process:

As we know the software development is a teamwork where every individual puts his/her efforts to produce quality software.

The personal software process (PSP) is a software engineering techniques that a software engineers learns and practices. It is proposed by Humphrey, Capability maturity model (CMM) is the development practices these activities are. Initial, Repeatable, Defined, managed and optimizing. Similarly PSP activities are planning, Design and design review. Coding and coding review, testing, postmortem.

* Team software process:

Team Software process is a technique to use the team skills to produce good quality software.

TSP is designed to facilitate superior performance of the software teams.

The main aim of TSP is everybody has a common understanding of the project plan and project schedule and project progress.

TSP activities are Launch, high level design, coding, testing, and postmortem.

PROCESS TERMINOLOGY.

Elements of software process.

1. Artifacts → small information i.e phase of project
2. Artifact sets → complete information of project i.e phases
3. Activity → Rules or procedures
4. Constraints → conditions
5. people → customer, enduser, stakeholder, Analyst, designers developer, tester, manager etc.
6. Tools and Technology → programing languages & compiler, editor
7. Method or Technique → Analysis, designing, coding and testing.
8. Relationship → communication among people
9. Workflow → framework activities
10. Milestones → status.

PRODUCT AND PROCESS.

→ A product is the outcome of a software project produced through process.

→ Customer needs converted into project.

→ The project uses process models and Resources to produce product.

Important Questions:

* 1. Define software engineering. What are the challenges of software engineering?

2. Explain briefly software development life cycle (SDLC)?
(Or)
Explain briefly prescriptive process model?

3. what are the software myths explain briefly?

4. What are the nature of softwares?

5. Explain briefly Unified process model (RUP)?

6. Explain briefly client server software engineering?

7. Explain briefly Unique nature of Web apps?

8. Explain about evolution of software development process models?

* Evolution of software engineering methodologies?

Ans: The stages of software engineering methodologies are

1. Exploratory methodology
2. structure oriented
3. Data structure oriented
4. Object oriented
5. component oriented

* Software crisis?

Ans: software crisis means the symptoms of the problems of software, i.e Economical problems.

* Agile process model?

Ans: There are some limitations of the existing process models, such as schedule slips, Business misunderstanding, defect rate, project failures, requirement changes, and so on. To overcome the above problems, the agile process model was introduced in 1990's. The agile process model is a group of software development methodologies based on iterative and incremental development.

* Characteristics of software and Hardware?

Software ① It is flexible ② It doesn't wear out
③ Failure rate is low compare than Hardware.
④ It is engineered (or) developed.

Hardware ① It is not flexible ② It does wear out
③ Failure rate is High ④ It is manufactured.

* Software engineering is a layered Technologies These are Tools, methods, process, quality focus