Date 21/7/19
Page No.

Program:- Set of instructions

Software - set of program + Documentation + Operation manuals

Software Engineering:- Engineering approaching to develop software

Q. What is Engineering? Diff betn arts, science & engineering.

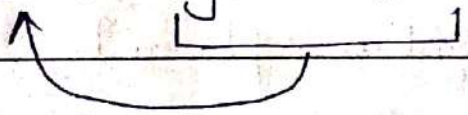
:) Deliverable vs Milestones

↳ milestone that can be completed & procured.

↳ checkpoints, points to reach ↳ requirements gathered completely

Project. Product-

:) ~~Product~~, ~~Project~~ & Process



*) 4Ps in SE:-

People, product, process, project.

↳ The way we produce product

↳ Process framework like Capability Maturity Model

↳ SDLC

* Software is:-

1) Instructions that when executed provide desired features.

Software types:

i) System software - OS, compile.
ii) Real time - Whether forecasting
iii) Embedded software - washing machine
iv) Webapps
v) AI software - expert systems.

* Software System vs System software ?

i) Software engineering is the establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines.

IEEE def:- Software engineering is application of a systematic, disciplined quantifiable approach to the development, operation and maintenance of software that is easy to use.

Date / /
Page No:

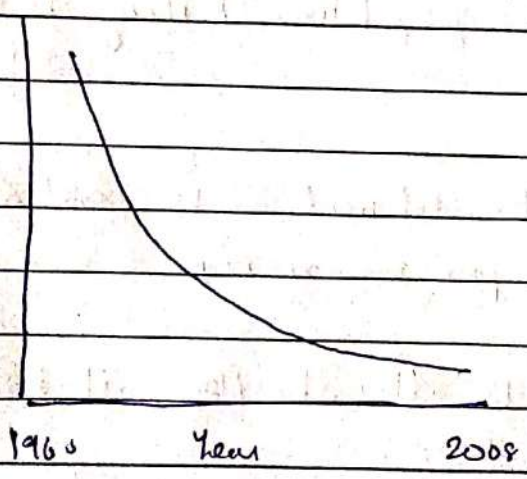
Why study SE?

i) Acquire skills to develop large programs.

→ Exponential growth in complexity & difficulty dues with size.

→ The ad hoc approach breaks down when size increases.

Software Crisis



Factors to the software Crisis:-

> Larger problems
> Lack of adequate training
> Increasing skill storage
> Low productivity improvements

High-level Language Programming (Early 60s)

- FORTRAN, ALGOL & COBOL introduced.
- Software development style was still exploratory.
  - Exploratory style - simple small programs with low complexity
  - Control Flow-Based Design (late 60s)

"Pay particular attention to the design of the program's control flow structure."

→ Using "Flow Charting Techniques".

Drawback

- go-to statement make control structure of a program messy.
- go to alter the flow arbitrarily
- restricted the use of goto

- Soon it was only three programming concepts were require
  - Sequence
  - Selection
  - Iteration

→ Called Structured programming methodology

and have modular design.

Modular — have entry & exit criteria

Interaction bet^{n} two modules is minimum.

Data Structure Oriented design (Jack Jon)

- Focus is on designing the data structure first
  ↳ derive program structure from it.

eg Jackson's Structured Programming (JSP)

Data Flow - Oriented Design (Jack Jon)

1. The data items input to a system must first be identified.

2. Processing required on the data items to produce the required outputs should be determined then.

) Selling point
) Simplicity

Smalltalk - Complete OOP
Java - Pure OOP

Date / /
Page No.

3) Object Oriented design (late 80's).

i) National objects occurring in the problem are identified

ii) Relationships among objects.

iii) Each object essentially acts as
- a data hiding (or data abstraction) entity.

OOP based
↑
Data flow based
↑
Data structure based
↑
Control flow based
↑
Ad-hoc

4) Aspect Oriented S/W Development
↳ Java + AspectJ

Mallad Sommerville

Date / /
Page No.

Plan-driven & agile process

The Software System

A structured set of activities required to develop a software system.

- Specification
  Design & implementation
  Validation
  Evolution

Plan-driven & Agile process

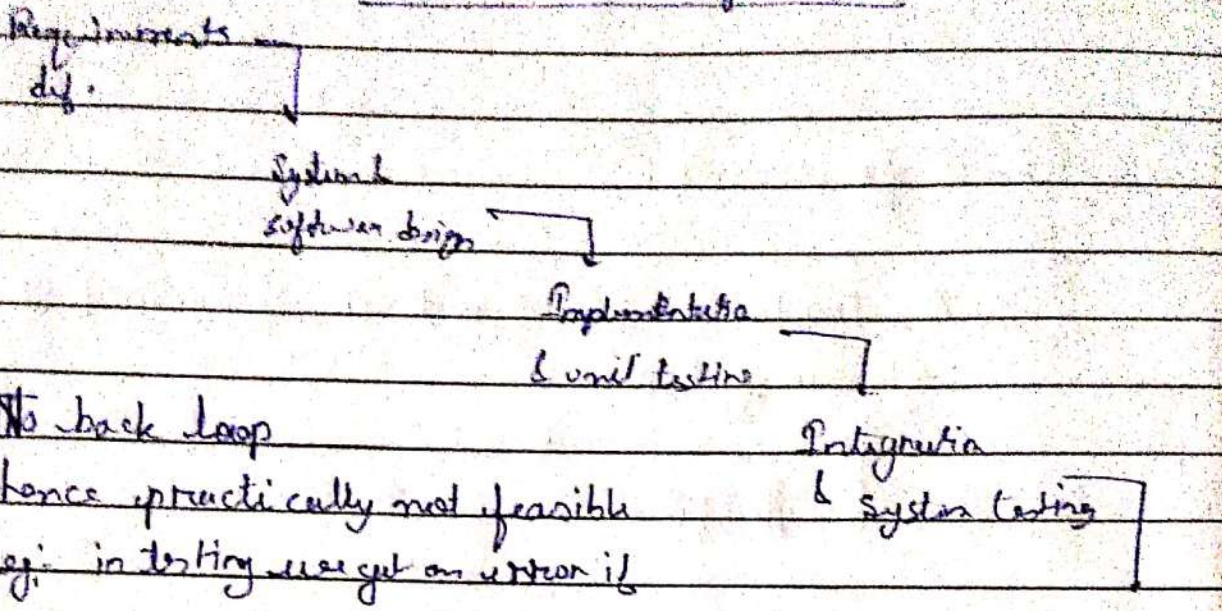
Software process models

1) The waterfall model
2) Incremental development
3) Integration & configuration

Plan-driven: All things are previously planned & software is designed according to plan & program is compared to that planned

Agile: Plan changes according to demands & requirements from customer.

Classical waterfall model

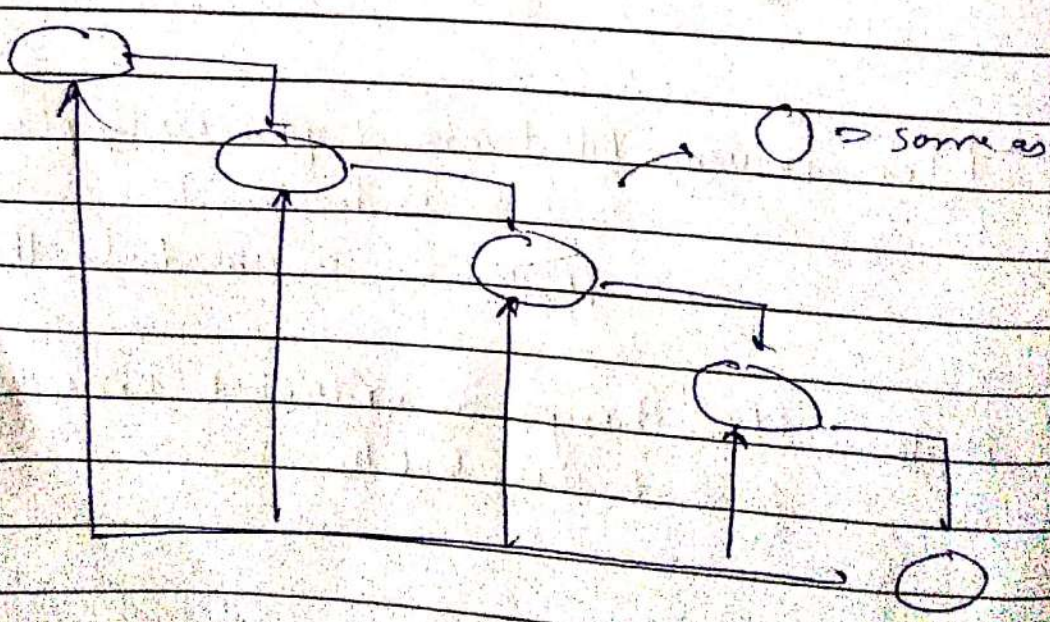


→ No back loop
→ hence practically not feasible
→ eg: in testing we get an error if we connect go to design phase then its of no use.

*) Among all life cycles — maintenance phase consumes max. effort

in development → 'testing' consumes max effort

Iterative waterfall model



Sommerville

Date / /
Page No.

Phase containment of errors

- Principle of detecting errors as close to its point of origin.

- Large project use ~~into~~ iterative waterfall model.

Incremental development:

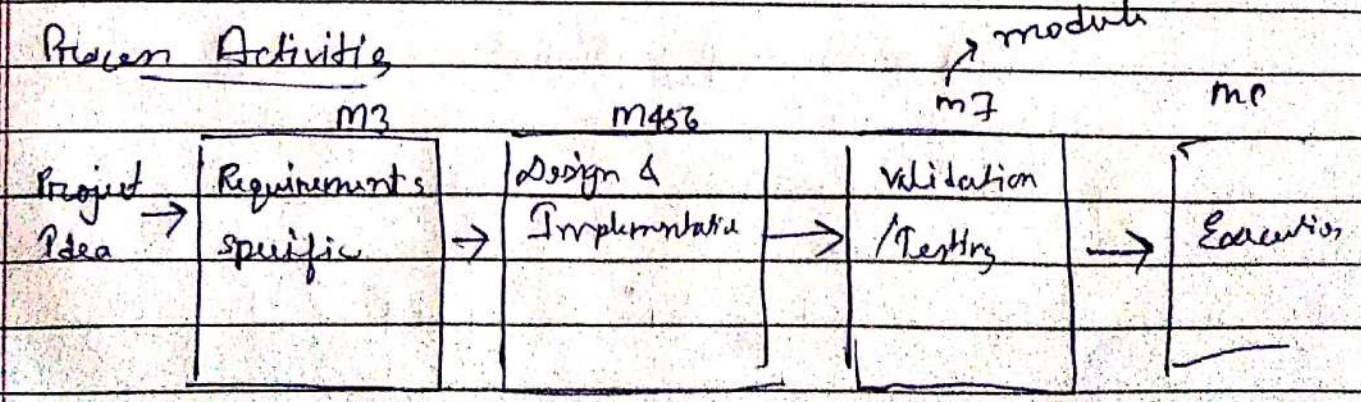
- New accomodation can be done.
- ~~But~~ A full-fledged plan is not made.

→ In this case process is not visible.
→ System structure tends to degrade as new increments are added.

Integration & Configuration development.

* Reuse-oriented software engineering.

Process Activities



Software specification / Requirements Engineering

↓

i) Requirements elicitation & analysis

ii) Requirements specification

iii) Requirement validation.

2) Software design & implementation
   - → Coding
   - → High level (see only the hardware & design. architecture)
   - → Low level (see the details of (detail design) requirements of design)

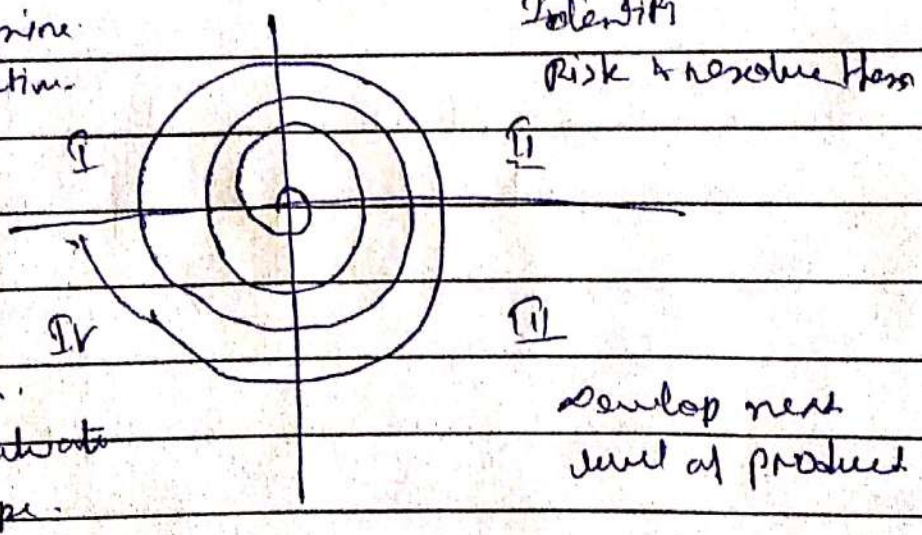
Design activity

Spiral Model (If we have higher amount risk involved).

Each loop of the spiral represents a phase of the software process:-

a) innermost loop:- system feasibility
b) next loop:- system requirements
c) next loop:- system design.

*) We do risk analysis after capturing system requirements.



→ bigger a part of spiral longer the duration.

the four parts are repeated.

Evolutionary model

Iterative :- complete requirements are first developed and SRS is finalized.

In evolutionary :- Requirements, plans evolve with time.

Agile Software Development :-

Effective (rapid and adaptive) response to change.

- Comm^n among stake holders
- drawing the customer onto the team
- Organized team so that it is in control of work performed.

Yielding :-

Rapid, incremental delivery of software.

Previous Jan

System implementation
& software validation

"V & V" :- - verification : Check if the model is correct or not
- validation : After the product is finished, the demand is matched with the customer needs.

Stages of Testing:

- V model.

Plan - driven - software process.

→ software evolution

Define system → Assess existing → Proven System → modify
evolution system changes

Q. Which model I should chose:

i) If I want to make an ERP system of NIT. & I don't have any past exp in soft dev.

ii) If I _____ iterative waterfall model
want to replace existing manual system.

iii) If _____
I want to existing automated system.

iv) If _____
I am expert in this

v) If _____
a client has no past knowledge.

Content:- Agile Software Development (Somerville)

↳ to move quickly (as we have a demand of change we quickly implement it to the ongoing process)

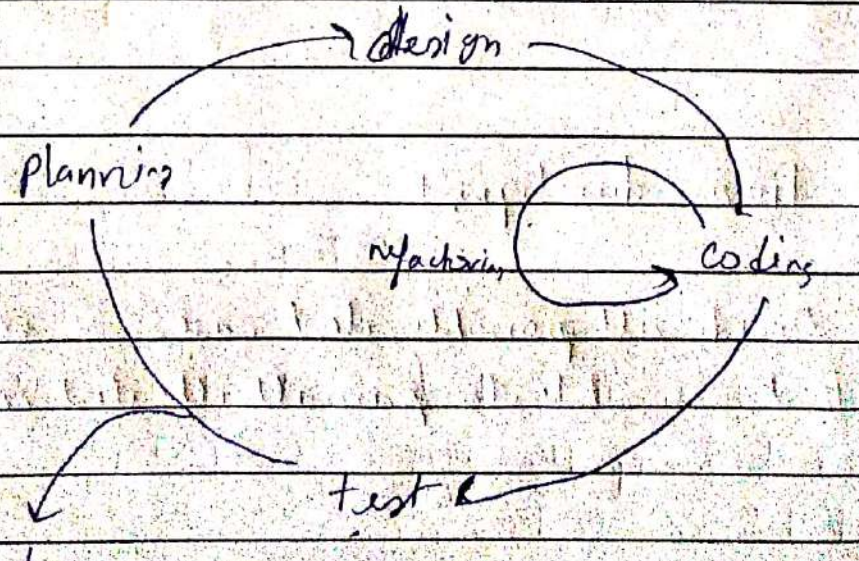
- 12 rules of Agile software development.

- Agile Manifesto

Principles:

i) Customer involvement
ii) Incremental delivery
iii) People not process (skill set of team)
iv) embrace change.
v) maintain simplicity.

Extreme programming process:-



features

- incremental planning.
- small releases.
- simple design
- test-first development
- refactoring.

→ pair programming (developers works in pairs. one codes other checks the code simultaneously).

→ collective ownership

→ Continuous integration

*) Refactoring:- Improving the software without concerning about application.

eg Re-organization class hierarchy to remove redundant code.

Test-driven development

i) Decide all possible test cases & make a software that is able to pass all the test cases.

(10²)² × 30 (3Rw)
10⁴ × 30

Distributed Scrum

Date / /
Page No.

*) Scrum sprint cycle.

↓

When all the developers, managers etc. are not located in a single location

*) Scaling up & scaling down

- Scaling up ↓ Using agile to develop large softwares.
- Scaling down ↓ using Agile in large organization

*) Practical problem with agile method.

*) Combining Agile + Plan Driven

*) Agile & Plan based Factor

- System
  - Scale
  - Type.
  - Lifetime
  - Regulation
- People & team
  - Competence
  - Distribution
  - Technology.
- Organizational issues
  - Contract
  - Delivery
  - Culture.

Requirement Engineering

#2

Functional requirements
Non-functional requirements
Domain requirements

Requirements Engineering Process.
↳ Four Key activities.

) Spiral view of requirements engineering.

) Requirements elicitation (& collection)
↳ Challenges — ?
i) client is not that clear about what he needs
ii) Communication gap.