Day: Fundamentals of SE. Date: / /

Chapter 1:-

(S.E)

- Software Engineering ⇒ Branch of computer science which deals with design, development, testing and maintenance of software applications.

- Software cost - more than hardware

Software Project failure

- Increasing System complexity ⇒ Advancement in S.E enable construction of larger, more complex systems and demand changes.

- Failure to use Software .E methods ⇒ Many companies skip software .E practices, leading to expensive and unreliable software.

- Software ⇒ Software that makes computer and phone work.

- good software ⇒ that is maintainable, dependable efficient, acceptable

- S.E activities ⇒
  - Software specification
  - Software development
  - Software evaluation.
  - Software validation

C.S	S.E
• understand how computers work / for solving problems • Programming languages / C.s solves problems	plan and manage software projects S.E makes practical and real-world software.

Day: Date: / /

System .E	S.E
• all aspects of computer-based systems development including software, hardware and processing.	It is general process coding / development.

- S.E costs ⇒ 60% - software development cost
  40% testing cost
  For custom, evolution cost often exceed development cost.
- Best SoE methods / techniques ⇒ There is no best method e.g., - games are developed by prototypes. We can't say, one method is better than other.
- Differences made to SoE ⇒ Web has revolutionized S.E by shifting it to internet-based applications. WoBs had led advances in software reuse / user-based.

Software Product:

Generic Products ⇒ general-system, sold to any customer, CAD Software

Customized Products ⇒ Specific-system, sold to specific customer, traffic monitoring systems.

Day: Date: / /

Essential attributes of good Software

- Acceptability ⇒ user - accept
- Efficiency ⇒ memory - utilization
  processing time
- Dependability ⇒ not cause physical and economic damage
  & Security • no external user should access or damage the system.
- Maintainability ⇒ • So, it could evolve to meet changing needs of customers.

General Issues that affect Software

- Heterogeneity ⇒ making same system that works on computers & mobile devices.
- Business and social change ⇒ Software that can be changed with economy and social changes.
- Security and trust ⇒
- Scale ⇒ that work from small gadgets to huge global systems.

Issues of professional responsiability.

- Confidentaility ⇒ encapsulation
- Competence ⇒ accept the work that you can do
- Intellectual property rights ⇒ copyrights and follow laws
- Computer misuse ⇒ don't mis-use

Day: Date: / /

Applications types:- (Bed sicket) (SMS)

✓ S -- Stand-alone application ⇒ GTA Vice City

✓ I -- Interactive transaction-based application ⇒ remote computer application, Web applications such as e-commerce applications. (e-commerce)

✓ E -- Embedded control system ⇒ Software systems that control and manage hardware devices. e.g. these are more (phone Wala)

✓ B -- Batch Processing Systems ⇒ Big data processors for business. and results in large group (Soap)

✓ E -- Entertainment System ⇒ personal use & entertainment

✓ S -- Systems for modelling & simulation ⇒ how to model physical situations

✓ D -- Data Collection System ⇒ collect data from environment using set of sensors and send to other system

S -- Systems of systems ⇒ systems that are composed of number of other systems.

Exclusive Falcon

Day: V-model (Waterfall - with verification and validation) Date: / /

Chapter 2:- (Software Processes) (S.P)

- S.P ⇒ same as S.E activities.
- S. Process descriptions ⇒ talking about their activities (UI, ~~data model~~)
  - Products (outcomes of Process activity)
  - Roles (People involved)
  - Pre & Post conditions (true before starting Program, true after)

⇒ Plan driven & agile processes (in advance)

plan-driven ⇒ We plan, than we do (progress is measured against plan)

Agile-process ⇒ Planning is incremental and easier to change (not for large scale project)

⇒ S. Process Models.

- Waterfall model ⇒ ② drawbacks:- one phase completed then move onto other. • Plan-driven (separate phases of specification and development) (③ large system engineering project.)
- Incremental development ⇒ (normal agile approach, Kuch beh phy kr loo) • Maybe plan-driven or agile ⇐ • Specification, development and validation are interleaved.
- Integration & configuration ⇒ • Maybe plan-driven or agile

Waterfall Model

① phases:- Requirements definition → System & Software design → Implementation & unit testing → Integration & System testing → operation and maintenan.

Day: COTS (commercial-off-the-shelf) Date: / /

Incremental development.

Specification ⇄ Initial version

↑ ↓

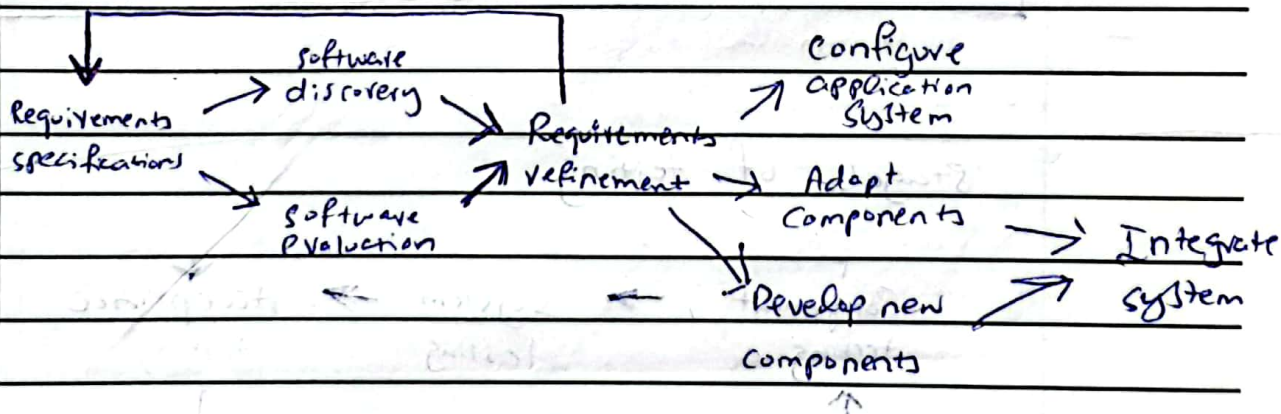
description → Development ⇄ Intermediate version

↑ ↓

Validation → Final version

- Waterfall cost high / Incremental cost less
- Customer feedback
- Fast delivery
- Process not visible.
- Reuse. (COTS)

Re-use-oriented Software Engineering



Process activities.

- Software specification → which services are required.
- Software design & implementation →

Exclusive Falcon

Day: Date: / /

general model of design process

Design inputs.

platform information	Requirements specifications	Data description

↓

Design activities

Architectural design (overall structure of system) → Interface design → Component design (search for reusable component)

Architectural design ↘ Database design (system data structure). ↙ Component design

↓

Design outputs

System Architecture	Database specification	Interface specification	Component specification

stages of testing

Component testing → System testing → Acceptance testing

(Acceptance testing loops back ↑ to Component testing)

- Software evolution → ~~Batch-processing system~~ Business and social change.

Day:

⇒ Copying with Change.

- Change must
- Chaning new cost and old wali ki beh cost

⇒ Reducing Colts of networks

- Phly hae wo work kr loo, baad ma change krnay kae zarroret hae nai.
- Design the procen in such a way that you can tolerate it.

⇒ Copying with changing requirements.

- System prototyping, make a rough draft and then change.
- Incremental delivery, give vlev to comment and experiment.

⇒ Software Prototyping

Establish prototype objectives → Define Prototype functionality → Develop prototype → Evaluate prototype

Day: Repeatly ↓ Increase. ↑ Date: / /

Iterative and Incremental Development

- High risk
- Product is required quickly/early
- easier to test or debug

Incremental development (team not trained)

- all requirements given.
- design complete product first
- only leave out which can be decided later.
- than slice it up and chunks
- can not go back to previous.

Iterative development (objectives clear).

(team trained)

- discovering what and how we need to go
- overall solution than refine some of areas.
- if created well, keep it. else discard it.

Incremental Delivery

- not in single delivery, broken down into increments
- user requirements are prioritised ✓

Incremental Delivery Advantages

- Customer value ↑
- lower risk of project failure

When to use Incremental ?

- Requirements are understood. ✓
- demand ↑
- team not trained.
- high risk features.
- web application and product based compapies.).

Exclusive Falcon

Day: Date: / /

Iterative Model

Design → test → implementation

- well design result.
- ① high level design
- than make project and than iterate.
- step by step improvement.
- more time on designing
- can get user feedback.

⇒ When to use iterative model?
- market ma jldi bhajna ho

⇒ Difference

	Incremental	Iterative
•	First specify, than it can not be Changed	we can discover them as build your project.
•	building separate parts	polish entire car without breaking it
•	ek dafa ban gya phr change nai	step by step

Day: Date: / /

⇒ Prototype :- (Customer can not define requirements early → we use prototyping)

Benefits ⇒ maintain ...

Prototype development ⇒ • focus on areas which are not well-understood.

•

Controlling Changes with prototyping ⇒ Cosmetic (35%)
local (60%)
global (5%)

Disadvantages ⇒ • Customer sometimes demand actual product after seeing prototype.
• costly.

⇒ Spiral Model (combine incremental and iterative).
• (does not work well for smaller projects)
• focuses on risk identification

⇒ Rapid Application Development (RAD)
• [redacted]
not agile method.