

Day:

# Fundamentals of SE

Date: / /

## Chapter 1:-

(S.E)

•- Software Engineering  $\Rightarrow$  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  $\Rightarrow$  Advancement in S.E enable construction of larger, more complex systems and demand changes.

•- Failure to use software S.E methods  $\Rightarrow$  Many companies skip software S.E practices, leading to expensive and unreliable software.

•- Software  $\Rightarrow$  Software that makes computer and phone work.

•- good software  $\Rightarrow$  that is maintainable, dependable, efficient, acceptable

•- S.E activities  $\Rightarrow$ 

- Software Specification
- Software development
- Software Validation
- Software evaluation.

### C.S

• Understand how computers work / for solving problems

• Programming languages / C.S solves problems

### S.E

plan and manage software projects

S.E makes practical and real-world software.

Exclusive Falcon

Day:

System - E	S.E
<ul style="list-style-type: none"> <li>all aspects of computer-based systems development including software, hardware and processing.</li> </ul>	<p>T<sub>1</sub> is general process coding / development.</p>

- S.E costs  $\Rightarrow$  60% - software & development cost  
40% testing cost

For custom, evolution cost often exceed development cost.

- Best soft methods / techniques  $\Rightarrow$  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~~ made to S.E  $\Rightarrow$  Web has revolutionized S.E by shifting it to internet-based applications. Web has had real advances in software / user-based.

Generic Products  $\Rightarrow$  general - system, sold to any customer, CAD Software

Customized Products  $\Rightarrow$  specific - system, sold to specific customer, traffic monitoring systems.



Day: / /

Date: / /

## Essential attributes of good software

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

## General Issues that affect Software

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

## Issues of professional responsibility.

- - Confidentiality  $\Rightarrow$  encapsulation
- - Competence  $\Rightarrow$  accept the work that you can do
- - Intellectual property rights  $\Rightarrow$  copyrights and follow laws
- - Computer misuse  $\Rightarrow$  don't misuse

Day:

Date: / /

## Applications types:-

~~Desktop~~  
(Bed side) (SMS)

- ✓ S • Stand-alone application ⇒ GTA Vice City
- ✓ I • Interactive transaction-based ⇒ Remote-computer application, web applications (e-commerce) such as e-commerce applications.
- ✓ E • Embedded control system ⇒ Software systems that control and manage hardware (phone Wala) devices. e.g., these are more
- ✓ B • Batch Processing Systems ⇒ Big data processors for business and results in large group
- ✓ 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.



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

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

- S.P  $\Rightarrow$  Same as S.E activities.
- S.Proces descriptions  $\Rightarrow$  talking about their activities  $\Rightarrow$  UML, data model.
- Products (outcomes of Process activity)
- Roles (people involved)
- Pre & Post conditions (true before starting Program, true after)

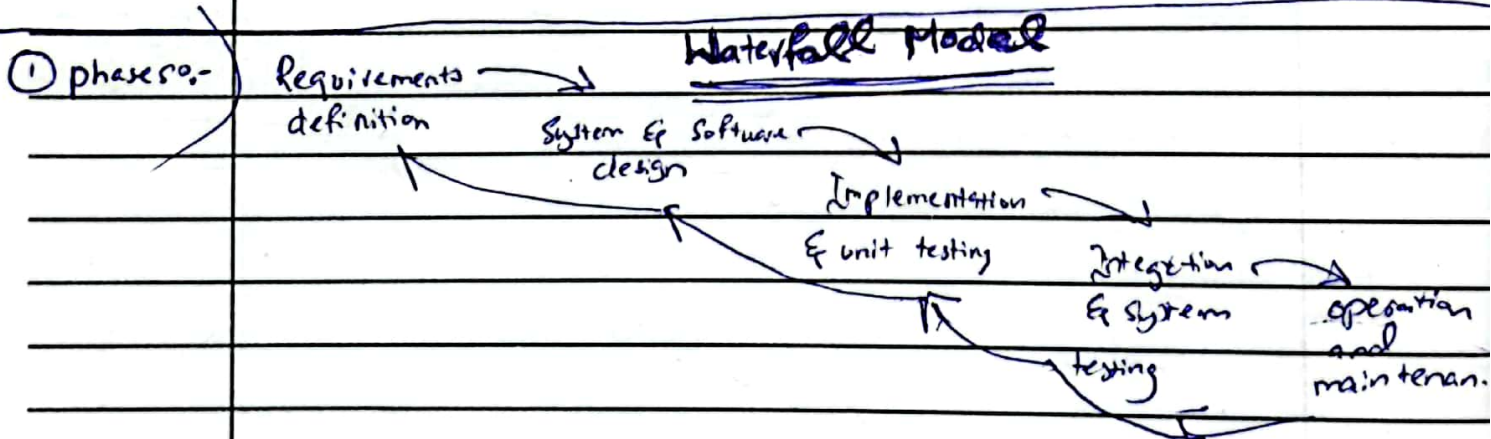
$\Rightarrow$  ~~Plan driven~~ & agile processes

plan-driven  $\Rightarrow$  We plan <sup>in advance</sup> then we do (progress is measured against plan)

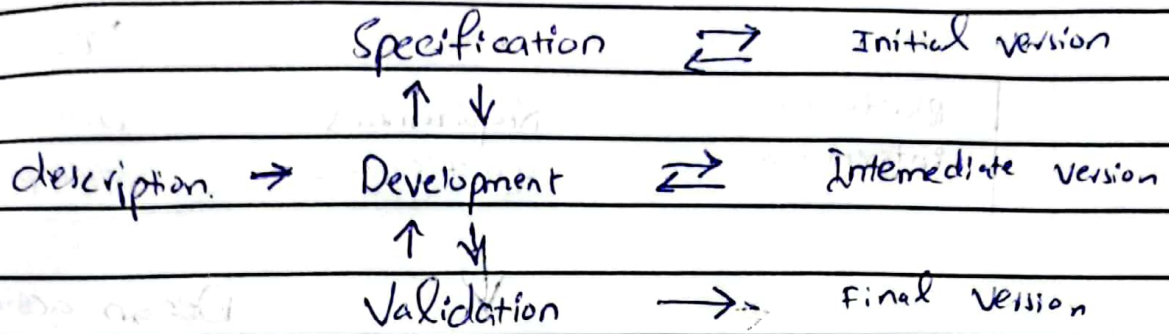
(not for large scale project)  $\Rightarrow$  Agile process  $\Rightarrow$  Planning is incremental and easier to change

$\Rightarrow$  S.Process Models.

- ③ large system Engineering Project.  $\Leftarrow$  ③ drawbacks: - one phase completed then move onto other.
- Waterfall model  $\Rightarrow$  Plan-driven (separate phases of specification and development)
- Incremental development  $\Rightarrow$  (normal agile approach)  $\Rightarrow$  Maybe plan-driven or agile
- Specification, development and validation are interleaved.
- Integration & configuration  $\Rightarrow$  Maybe plan-driven or agile



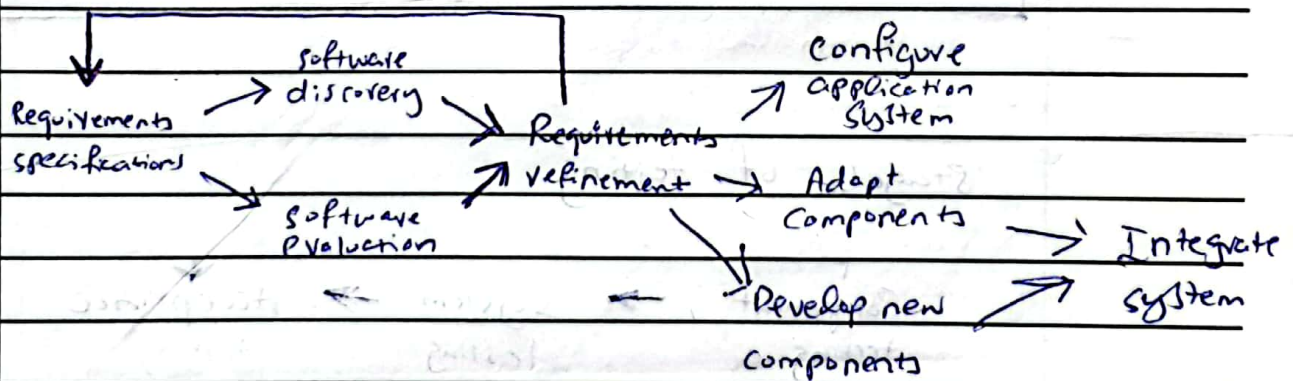
## Incremental development



• Waterfall cost high / Incremental cost less

- Customer feedback
- Fast delivery
- Process not visible
- Reuse (COTS)

## Re-use-Oriented Software Engineering



## Process activities

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





⇒ Copying with change.

- - change mult.
- - Changing new cost and old wali ki beh cost

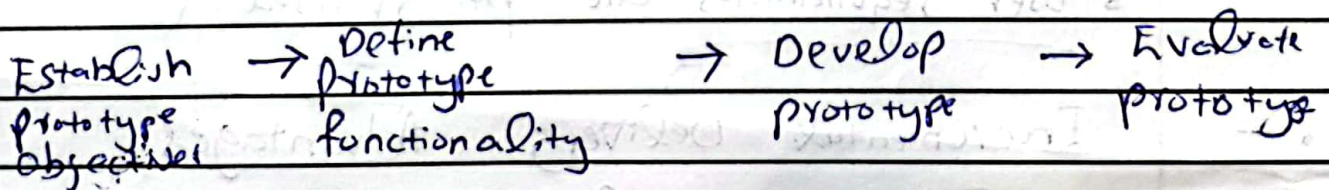
⇒ Reducing Costs of network

- - phly hae wo work kr loo, band ma change krnay kae zarorat hae nas.
- - Design the process 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 user to comment and experiment.

⇒ Software Prototyping





Day:

repeatedly  
↓

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.
- then slide it up and chunks
- can not go back to previous.

team + trained

Iterative development (objectives clear).

- discovering what and how we need to go
- overall solution then 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 companies.

Exclusive Falcon

Iterative Model

Design → test → implementation

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

⇒ When to use iterative model?

- market is highly changeable

DifferenceIncremental

- First specify, then it can not be changed
- building ~~separate~~ separate parts
- ek data kar gaya phir change nai

Iterative

- we can discover them as build your project
- polish entire car without breaking it
- ~~step~~ step by step



Day:

Date: / /

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

Benefits ⇒ maintain ...

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

Controlling Changes with prototyping ⇒ Colmetic (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)

~~Agile software~~  
not agile method.