Lec 2 SDLC

Date No

Software Development Life Cycle

What is an engineering process.

> in engineering, a process is a set of interrelated tasks that, together transform input into output

What is a software development process?

> A set of Activities and Associated Results that produce a software.

Fundamental Process

1. Software Specification
2. Software Development
3. Software Validation
4. Software Evolution

Date No

~~Standards~~

1. ~~Confidentiality~~
2. ~~Competence~~
3. ~~Intellectual property rights~~
4. ~~Computer misuse~~

(1) Software Specification

> Documenting
>
> → Expectations on the system.

> Lays out requirements

- Written & Diagrammatic

→ Description of the services that the future system must provide.

Date No

(2) Software Development.

> ~~Software development involves designing and implementing the~~

> Software development

designing & implementing

the system

according to the specification.

(3) Software Validation

> Software validation involves

checking & ~~ref~~ verifying

whether,

the system fulfills the requirements.

Date No

(4) Software Evolution

> Software needs to

change & upgrade

↓

with time

Software development evolution

> waterfall

> Agile

> DevOps

(*) What is a SDLC?

> SDLC is a framework that defines activities performed

↓

throughout the software development process.

Date No

Life Cycle Model (Process Model)

> A software life cycle (process) modal :

↳ Descriptiv and diagrammatic model of the life cycle of a software product.

↳ identifies all the activities and phases necessary for software development.

↳ establishes a precedence ordering among the different activities.

* Life cycle models encourage systematic and disciplined software development.

Date No

Life Cycle Models

Traditional Approaches

1. Waterfall Model
2. Incremental Model
3. Prototyping Model
4. Spiral Model
5. Unified Process

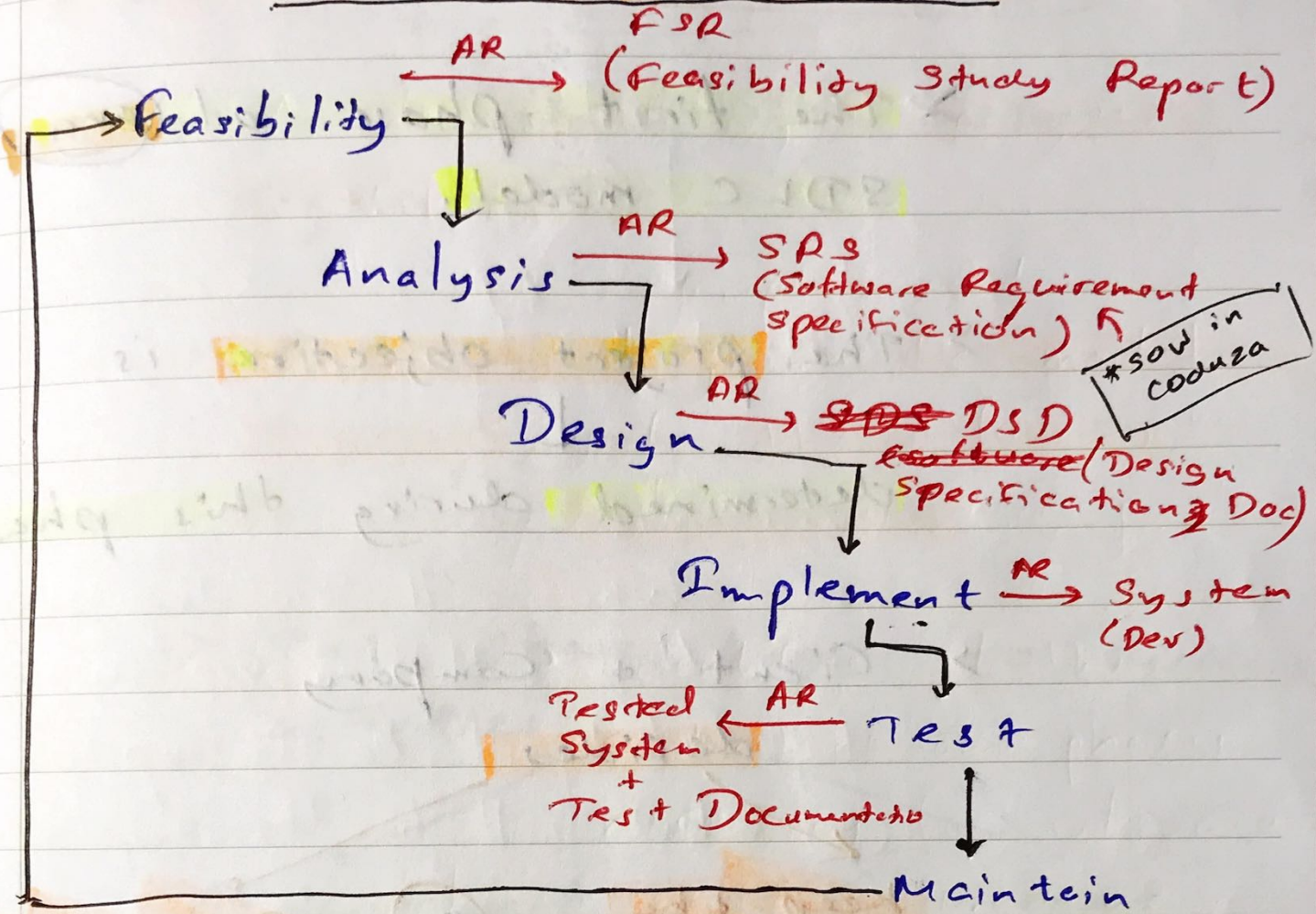
Modern Approaches

1. Agile Methods
2. Secure software Development

1

Date No

Waterfall Model



> The most well known SDLC.

> Simple → understand / use.

> Each phase begins → (only) after the previous phase is over.

> Also called Linear Model

> A document driven process

> This model specifies what the system is supposed to do before building the system

Date No

① Feasibility Study

> The first phase of any SDLC model.

> The project objective is determined during this phase

> Client & Company decides

- Keep the existing system (or)
- Build a new software

in this stage

Why do a feasibility study?

To provide management with enough information to know:

1. Timeline
2. Worth? (Beneficial?)
3. Alternative solutions
4. Whether there's a preferred alternative?

Date No

② The Requirements Phase

Aim: to understand the customer's requirements.

> A customer may be a single person, a group, ~~pg~~ a department, or an entire organization

> This phase have two distinct activities

① Requirement Gathering & Requirement Analysis

② Requirement specification

Date No

2a.1 Requirement Gathering

> Stakeholders

↓ to find out

"What to be done"

> Questions:

1. Who will use the system?
2. How will they use the system?
3. What will be the inputs?
4. What will be the outputs?

Requirement Gathering: meetings, interviews, discussions.

Date No

29.2 Requirement Analysis

Aim: To understand exactly what the customer needs.

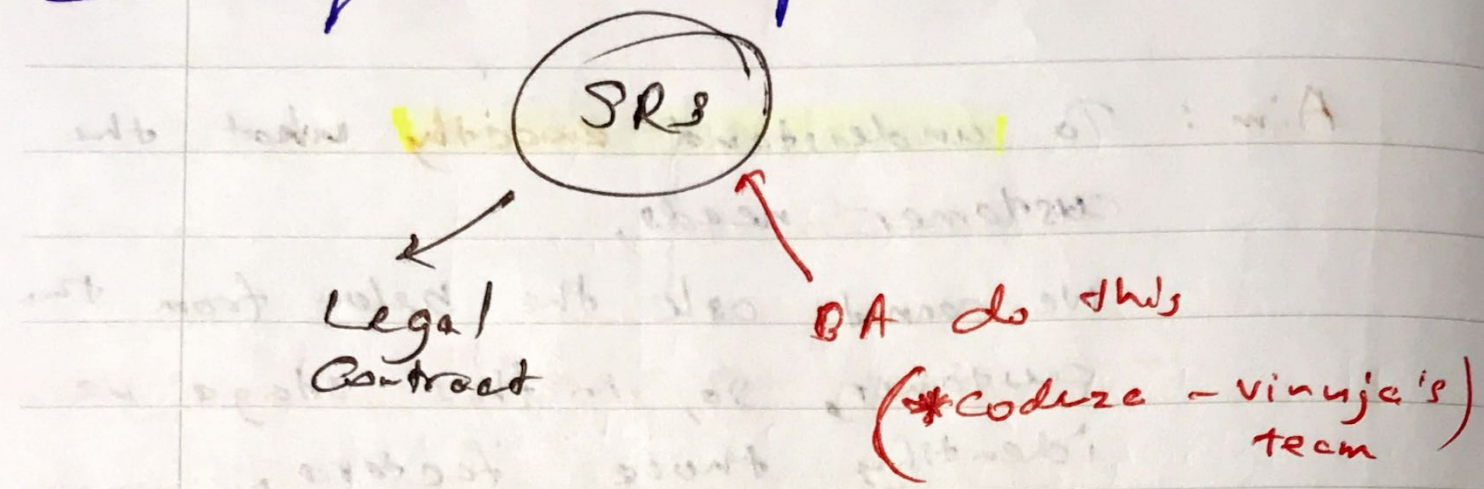
We cannot ask the below from the customer, so, in this stage we identify these factors.

1. Inputs (data) to the system.
2. Processing to be performed.
3. Outputs from the system.
4. Characteristics of the system.
5. Constraints on the system

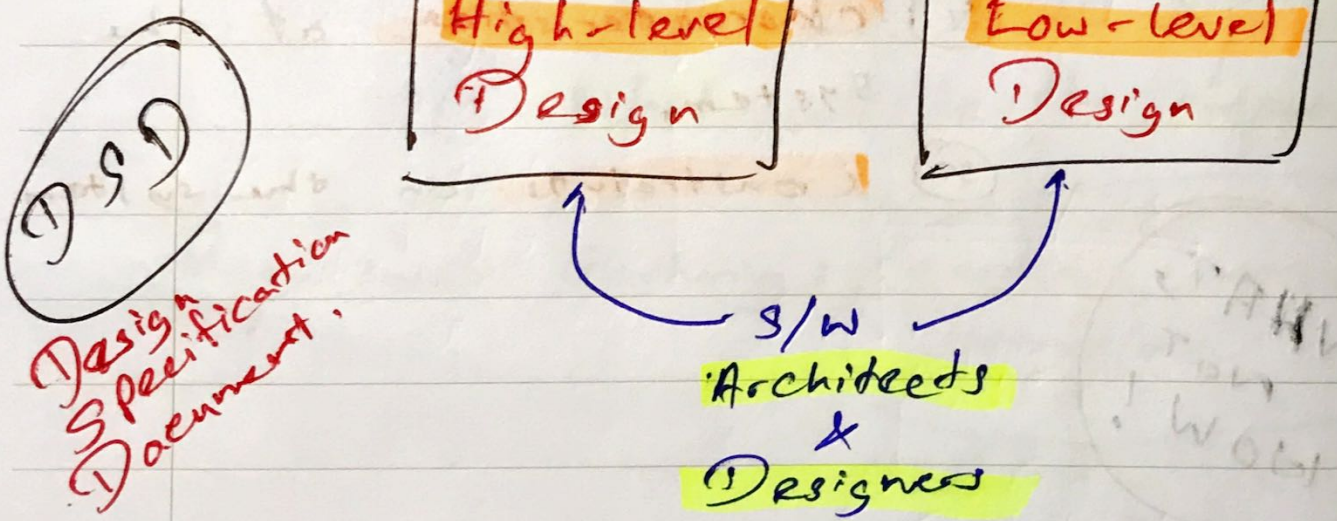
WHAT, NOT HOW!

Date No

2b Requirements Specification



3 Design



> Decisions are made about,

① Hardware (ex: POS machines & etc)

② Software (ex: POS S/W)

③ System Architecture (arr syst, Pen)

Date No

(4) Development

> on receiving system design documents, the work is divided in → modules

Specific component

Sprint → Specific time

> Dev Team → as per DSD → Code the S/w (chosen programming lang)

This lang chosen in the design phase.

Some

> Testing → Discover faults → Debug

Date No

5 **Testing**

* The testing phase
  - ensure
    - that the software requirements are in place } and → that software works as expected.

* Defect Identify —Tester inform→ Developer

* If defect valid?
  - Yes → Developer
    - resolve
    - a new version

* The above cycle continues until all defects are mitigated and the S/W is ready for deployment into the Production environ

Date No

(6) Deployment and Maintenance.

> Once the software is error free

↓

It is deployed

↓

into the operating environment.

> While the customers are using the software

→ any issue will be brought to the attention of the maintenance team.

> They work to resolve them immediately

Date No

Summarized Flow

① Feasibility Study

* project objective identify

Feasibility study ① Doc

- Technical Feasibility
- Economic Feasibility
- Schedule Feasibility
- Operational Feasibility
- Legal & Regulatory Feasibility

* initial decision

② The requirement phase

① 2.1 a Requirement Gathering

Meetings (stake holders) (client)

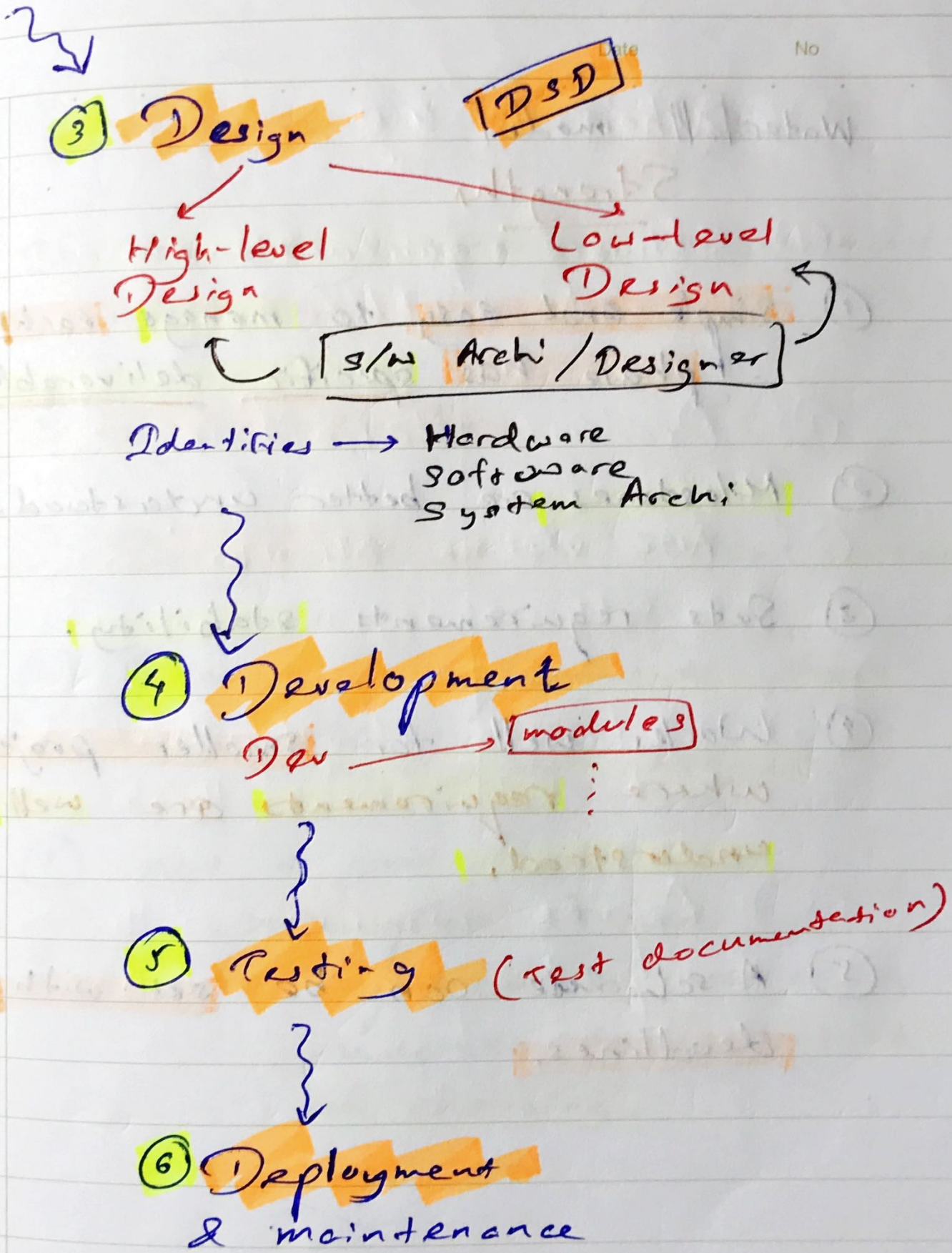
② 2.1 b Requirement Analysis.

- ① Decides
  - > Inputs / outputs
  - > Processing
  - > Characteristics
  - > constraints

Who what Input/output

② 2.2 Requirement Specification

SRS 3.4



Date No.

Waterfall model

Strengths

1. Simple and easy to manage each phase has specific deliverables.
2. Milestones are better understood.
3. Sets requirements stability.
4. Works well for smaller projects where requirements are well understood.
5. A schedule can be set with deadlines.

Date No

Weakness

1. No working software is produced until end.
2. High certainty.
3. Delay discover of serious errors.
4. After the requirements phase
   ↓
   No way to change
5. Not a good model for
   - > Complex Projects
   - > requirements are at a moderate to high risk of changing.

Date No

When to use?

1. Software requirements clearly defined and known
2. Product definition is stable
3. New version of the existing Software system is created.
4. Software development technology and tools are well known.
5. Ample resources with required expertise are available

(2)

Date No

Iterative Model

> The classical waterfall model is idealistic:

(classical one)

assume that everything will be okay

but defects can be introduced during any dev stage.

↑

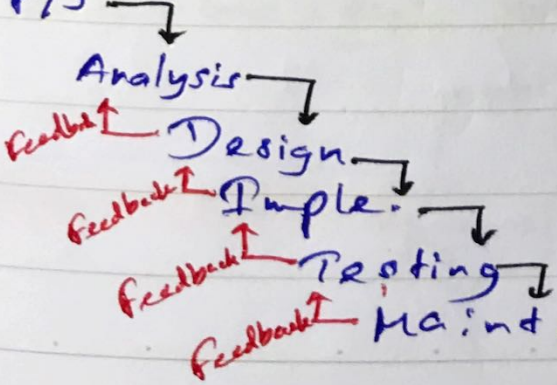
In practice

Hence feedback paths must be added to the → classical waterfall model.

↓

The resultant:

Iterative Waterfall Model



Date No

Iterative waterfall model

strengths

1. Defects are detected and fixed early through the feedback path.

weakness

1. Limited customer interaction
2. Difficult to incorporate change requests.

(3) Date No

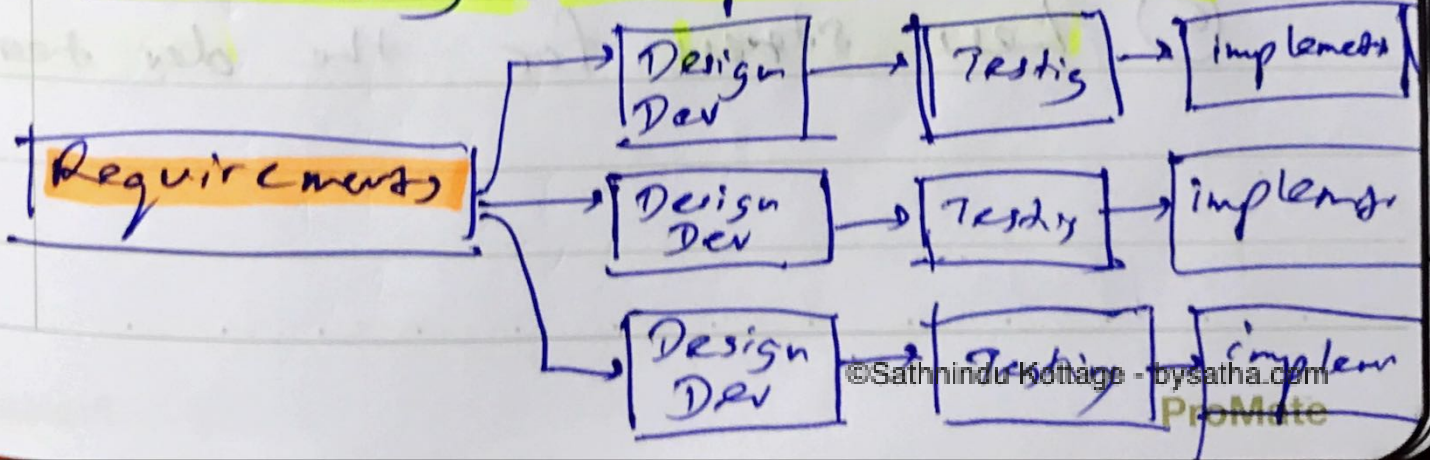
Incremental Model

> Incremental model is an evolution of waterfall model.

* The product is designed, implemented, integrated and tested as a series of incremental builds.

> The incremental model prioritizes requirements of the system and then implements them in groups.

> It is the process of constructing a partial implementation of a total system and slowly adding increased functionality or performance.



Date No

Incremental Model

Strengths

1. Generate working software quickly and early during the software life cycle.
2. More flexible - less costly to change scope and requirements.
3. Easier to test & debug.
4. Easier to manage risk.
5. Lower initial ~~cost~~ delivery cost.
6. Less stress for the dev team.

Date No

Weakness

1. Requires good planning and design.
2. Each phase of iteration is rigid and do not overlap each other.
3. Demarcation of increments can be difficult in a practical application
4. Total cost of the system might not be lower.
5. Problems may arise pertaining to system architecture because not all requirements are gathered up front for the entire software life cycle.

Date No

When do use?

1. on projects which have lengthy development schedules.
2. A need to get basic functionality to the market early.
3. Most of the requirements are known up-front but are expected to evolve over time.
4. On a project with new technology.

4. Date No

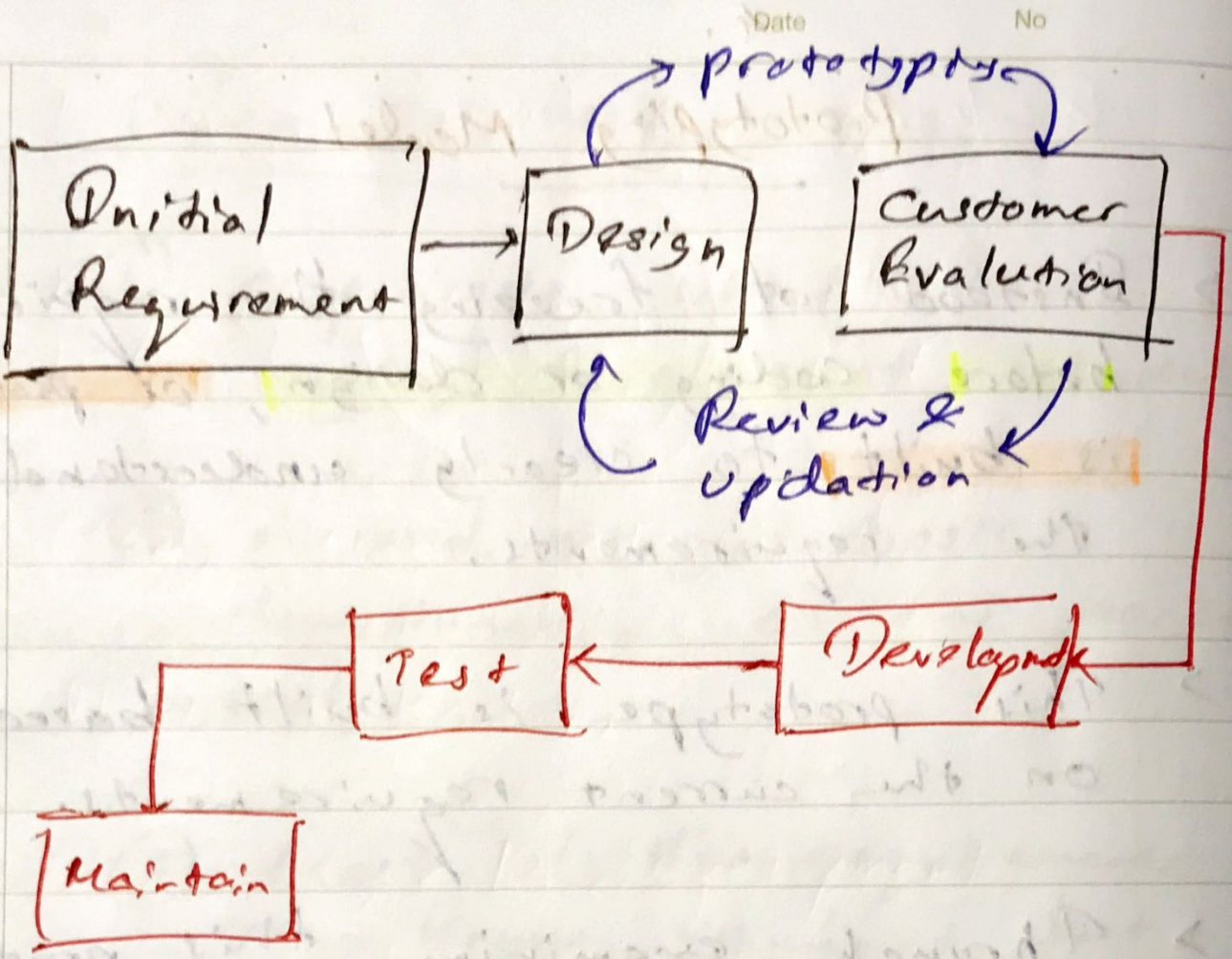
Prototyping Model

> Instead of freezing the requirements before coding or design, a prototype is built to clearly understand the requirements.

> This prototype is built based on the current requirements.

> Through examining this prototype, the client gets a better understanding of the features of the final product.

> Requirements may be changed with the client feedback on the prototype.



Prototyping Model

Strengths

(1) Ability to clarify user's expectation for the system to be developed.

(2) Prototype simulates awareness of additional needed functionality.

(3) Better user satisfication

Date No

④ Early user feedback.

Weakness

① Scope creep - The system scope may expand beyond original plans.

② Overall maintainability may be overlooked

③ The customer may want the prototype be delivered

④ Process may continue forever.

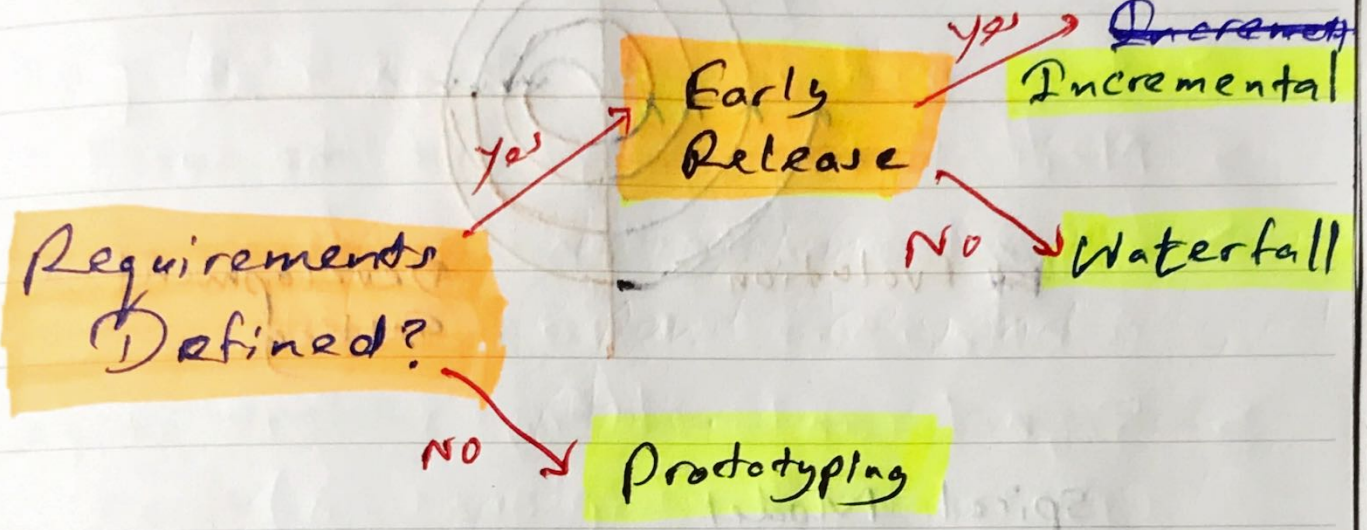
Date No

When to use Prototyping

1. Requirements are unstable or have to be clarified
2. Many user interfaces
3. New technology
4. New, original development
5. Developers are not familiar with the technical and development tools.

Date No

Selection of Approach



5. Spiral Model

(1) Meta model - combines iterative and prototype dev with the systematic controlled aspects of the waterfall model

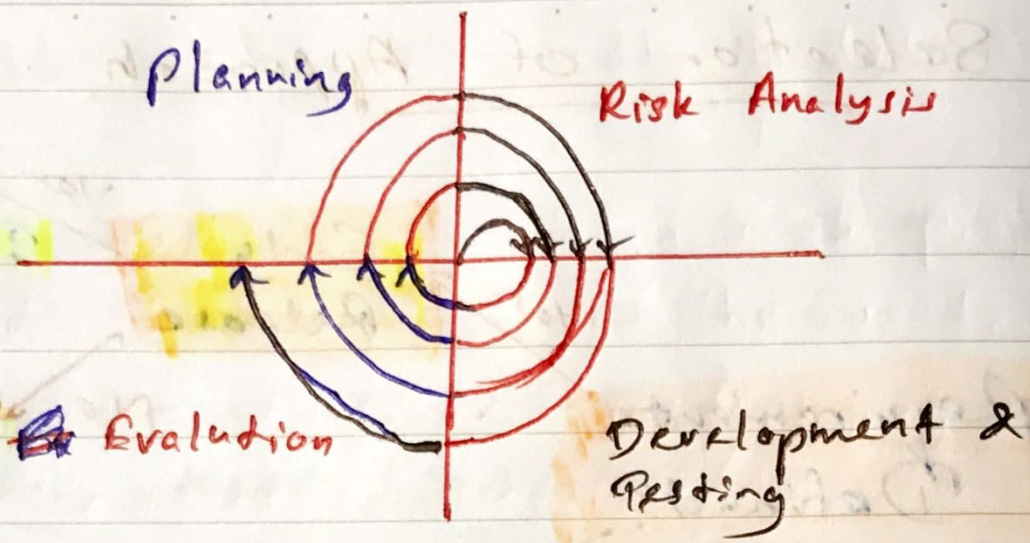
(2) Allows for incremental release.

(3) Introduced by Barry Boehm in 1986.

(4) Allows elements of the product to be added in when they become available or known.

(5) Emphasises on risk management.

Date No



Spiral Model

Quadrant 1: Planning

> Determine objectives, alternatives and constraints.

Quadrant 2: Risk Analysis

> Evaluate alternatives, identify, resolve risks.

Quadrant 3: Development & Test

> Develop, verify, next-level produc

Quadrant 4: Evaluation

> Analyse feedback and plan next phases.

Date No

Spiral Model -

Strength

> Focus on risk analysis
> Good for large and mission critical projects.
> A working software is produced early.
> The design does not have to be perfect.
> Early and frequent feedback from users.
> Cumulative costs assessed frequently.

Weakness

> Can be a costly model to use.
> Risk analysis requires expertise.
> Success is highly dependant on the risk analysis phase.
> Doesn't work well for smaller projects.

Date No

When to use Spiral model

- For medium to high-risk projects
- New technology to be used.
- Complex, constantly changing and continuous requirements.
- Significant changes are expected (research & exploration)
- Users are unsure of their needs.

Summarized Table

Unclear User Requirements	① Prototyping ② Spiral (exc) ③ Iterative / Incremental ④ Agile (exc)
Unfamiliar Technologies	① Prototyping (exc) ② Spiral (exc) ③ Iterative / Incremental

Date No

Complex System
1. Waterfall
2. V-shaped
3. Prototyping (exc)
4. Spiral (exc)
5. Iterative / Incremental

Reliable System
1. Waterfall
2. V-shaped
3. Prototyping (exc)
4. Spiral (exc)
5. Iterative / Incremental

Short time Schedule
1. Prototyping
2. Iterative / Incremental (exc)
3. Agile (exc)

Strong Project management
1. Any framework -excellent-

Date No

Cost Limitation
1. Iterative / Incremental (exc)
2. Agile (exc)

Skills Limitation
1. Waterfall
2. v-shaped
3. Iterative / Incremental
4. Agile (exc)

Documentations
1. waterfall (exc)
2. v-shaped (exc)
3. Prototyping
4. Spiral
5. Iterative / Incremental (exc)

Compository reusability
1. Waterfall (exc)
2. V-shaped (exc)
3. Iterative / Incremental (exc)
4.