Software Engineering

Software development life cycle (SDLC)

A. **Software Process Model / Lifecycle model** — Series of steps to be followed to develop software product

→ PHASES / STEPS

→ MODELS / Software development paradigm.

A strategy that defines SDLC

PHASE/STEPS

Communication

Requirement gathering & analysis

Feasibility analysis/study

~~Requirement analysis & specification~~ (System analysis)

Software design

Coding & unit testing

Testing (Integration)

Maintaince

Disposition

MODELS

Waterfall
V- Model
Prototype model
Evolutionary model
Iterative model
Spiral model
Agile model
RAD model

Tutorialpoint, wikipedia, javapoint (SDLC)

B. **Software project management (SPM)**

- Project is collection of well-defined task
- Has a distinct goal.

Project management is the undertaking the responsibility of executing software project.

Eg. Monitoring progress & performance, Risk analysis etc

- Project Planning
- Project estimation
- Project Scheduling
- Resource management
- Risk Management
- Monitoring
- Control
- Configuration

Project management tools

C. **Software quality management (SQM)**

Undertaking the responsibility that software product meets the quality expected by the customers.

→ Software quality assurance

CMM, ISO, Sigsigma

Phases/steps of software development life cycle (SDLC)

1. Communication by users

User initiates request for desired software product.
Submits request to software developer/service provider.

2. Requirement gathering & analysis

UNDERSTANDABLE, CONSISTENT, UNAMBIGUOUS
COMPLETE

Software development team makes discussion with ~~customers~~ customers and tries to bring out information on their requirements.

Inconsistencis and incompleteness in this requirements are removed.

(a part contradicts with other part)

some part of the requirement is missing

When requirements are properly understood, requirement specification activity starts ——— i.e organize requirements into SRS document (Software Requirement Specification)

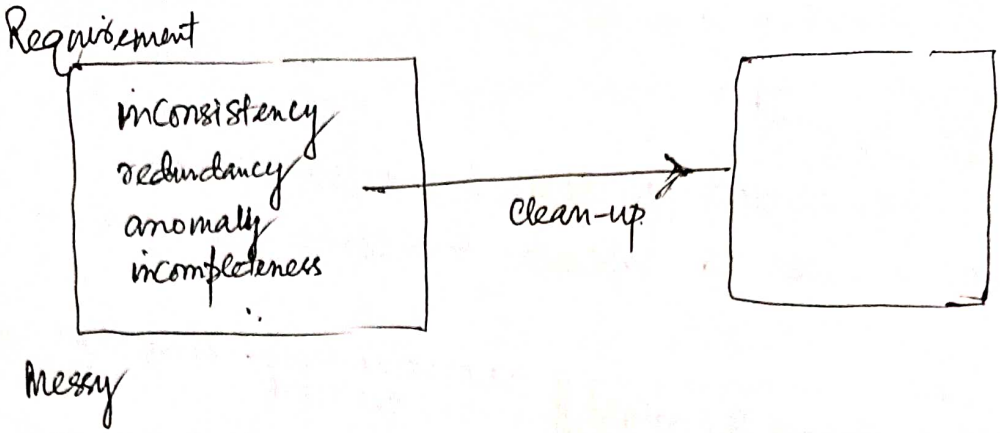
Improperly documented requirement increase number of iterative changes + push-up development cost + set up ground for better customer developer dispute and legal battle. ——(Need for SRS)

The engineers who gather and analyze customers requirement and write SRS are known as system analyst.

Once SRS is ready it is reviewed internally by the project team & then given to customer for review. When customer approves it, future development activities may start.

SRS document contains Functional requirement, Non-functional requirement and goals of implementation.

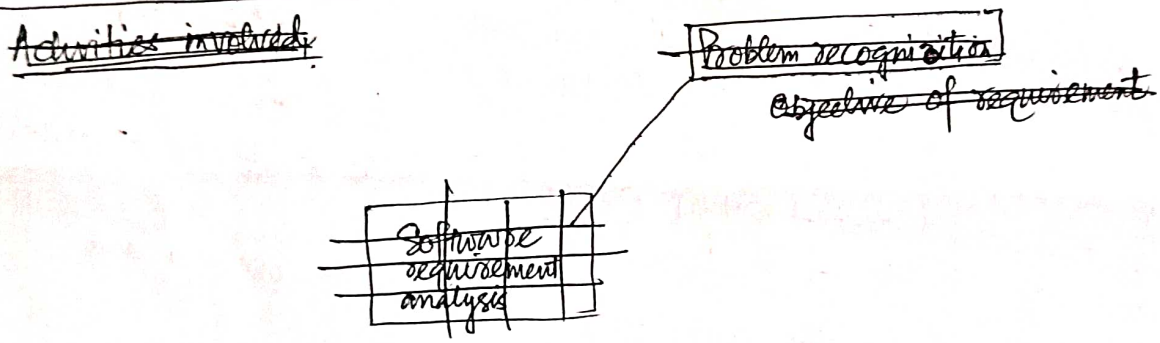
- Functional requirement: Set of functions performed by the system.
- Non-functional requirement: Characteristics of system cannot be expressed in terms of function eg. maintainibility, portability usability etc.
- Goal of implementation: General suggestions regarding development.



through interview.

collect
gather
examine

clean-up.



2. **Feasibility analysis** / Feasibility study

Determine if it is financially or technically feasible to develop the product.

- Formulation of Abstract problem defination + important requirements are taken & rest of the requirements are ignored.
- Formulation of different solution stategies.
- Compare benifits & shortcommings in terms of resource required, cost of development, time of development etc. based on high level design of the problem.

If none of the solutions are feasible then the project is abandoned.

Otherwise, → Best solution is picked for development.

Feasibility study gives conclusion whether to go ahead or stop.
This also helps to identify risk factors involved in.
Enhances success rate.

Need for FS

Types of feasibility study

1. Technical feasibility:
   technical skills, capability of technical team, technology used, ...
2. Operational feasibility
   How much easy will be the product development
3. Economic feasibility
   What will be the cost of development.
4. Legal feasibility
   Existance of legal barrier
5. Schedule feasibility
   How much time is required to complete the project

3. **Software design**

Transform requirements into suitable forms which helps programmer to implement it. In software design we consider the system to be set of components or modules with clearly defined behaviours and boundary.

Purpose of design.

Correctness	Flexibility
Completeness	Consistency
Efficiency	Maintainability (by other designers)
understandibility	

Design approaches.

- Function oriented (viewed as collection of functions)
- Object oriented (viewed as collection of objects)

4. **Coding** & unit testing

Translate software ~~design~~ design into source code. End product of this phase is set of program modules. To enable the engineers to write good program, every software development organizes normally formulates own coding standards. Coding standard involves code template, commenting guideline, variable & function naming etc.

After coding phase each module is unit tested to ensure correct working of all individual modules.

After a module is successfully compiled (i.e elimination of syntax errors), Code review is carried out. Purpose of code review is to find algorithmic & logical errors.

Code review

1. Code walk through : Some members of development team are given the code for reading & understanding.

6. **Maintainance**

Software is updated according to changes taking place in user environment or technology.

Types of maintainance

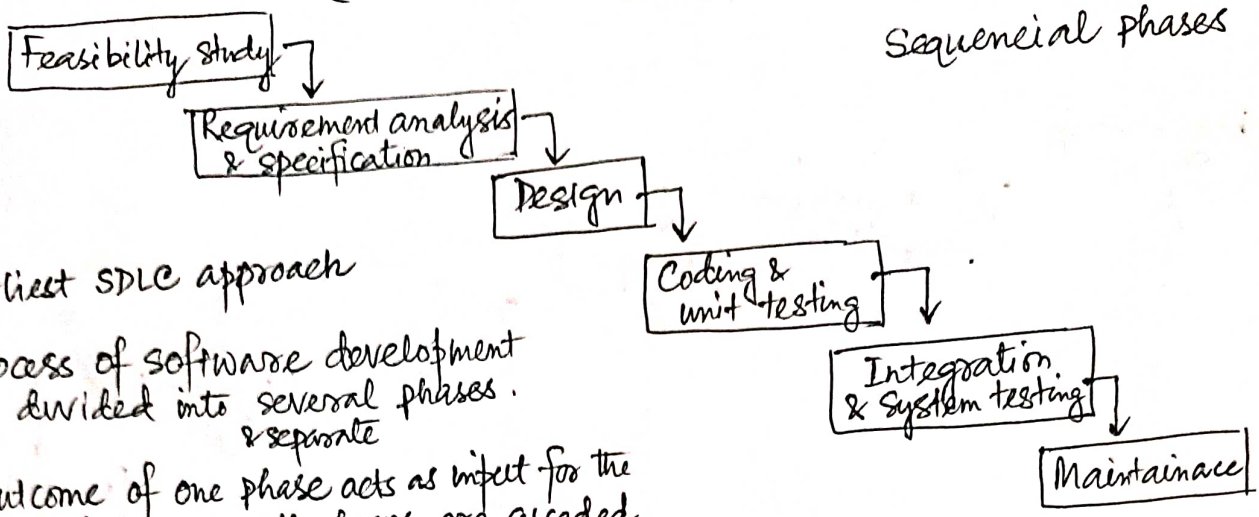
- Corrective: To rectify bugs observed while system is in use.
- Adaptive: Customise used product on new platforms ie new operating system or interfaced with new hardware.
- Perfective: Change functionality according to customer's demand.

7. **Disposition** As time elapses software may decline on performance. It may go completely obsolete or may need intense upgradation. Developer decides to close down the system.

WATERFALL MODEL (classical)

Linear sequencial life cycle model

Sequencial phases



Earliest SDLC approach

Process of software development is divided into several phases & separate.

Outcome of one phase acts as input for the next phase —— all phases are cascaded to each other.

No phases do not overlap.

This model is applicable when

Requirements are well documented, clear & fixed.

Technology is not dynamic

Project is short.

Management provides enough resources

Advantage

Simple, Easy to understand

Easy to use

Each stage is clearly defined

Process, Action & Results are well documented

Disadvantage

High amount of risk & uncertainity

Not good for complex project

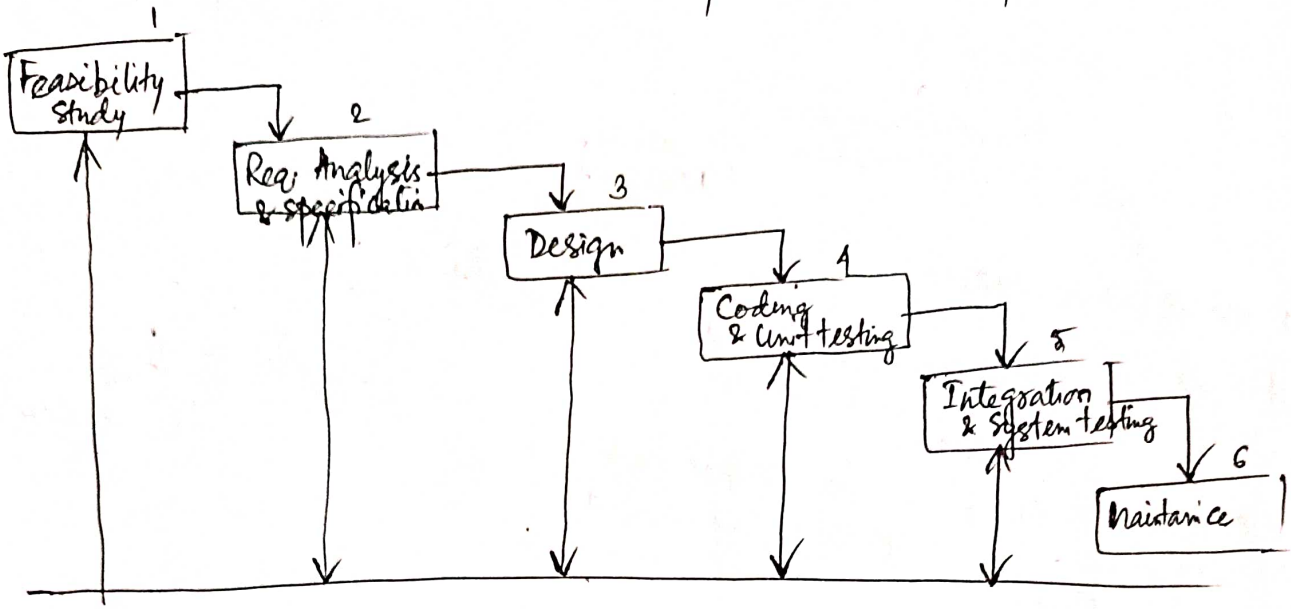
Very rigid: Cannot accomodate change

No error correction mechanism.

New phase starts only after previous stage ends —— efficiency reduces.

Iterative waterfall model

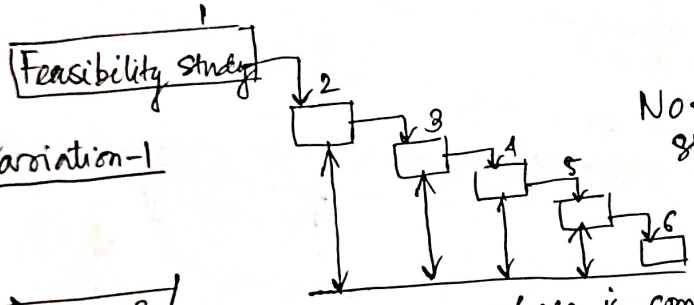
Extension of classical waterfall model.



Feedback path from every phase to its previous phase

When errors are detected at some phase, feedback path allows correcting errors.

Variation-1



~~Feasibility study~~

No feedback path to Feasibility study — once the project is taken does not give up project easily.

Next phase begins when previous phase is completed.

Applicable when

Requirement is ~~refined~~ & clearly understood.

New technology is learnt by development team.

More costly

Advantage

Feedback path allows correcting errors

Simple to understand & use

Customer involvement is not required during software development

Suitable for comparatively large & complex project

Disadvantage

To scope for any incremental delivery

No overlapping of phase

Has no risk handling mechanism

Limited customer interaction / review / change request

Customers interaction occurs at the start of the project & end of the project. Fewer customers interaction may lead to many problem.

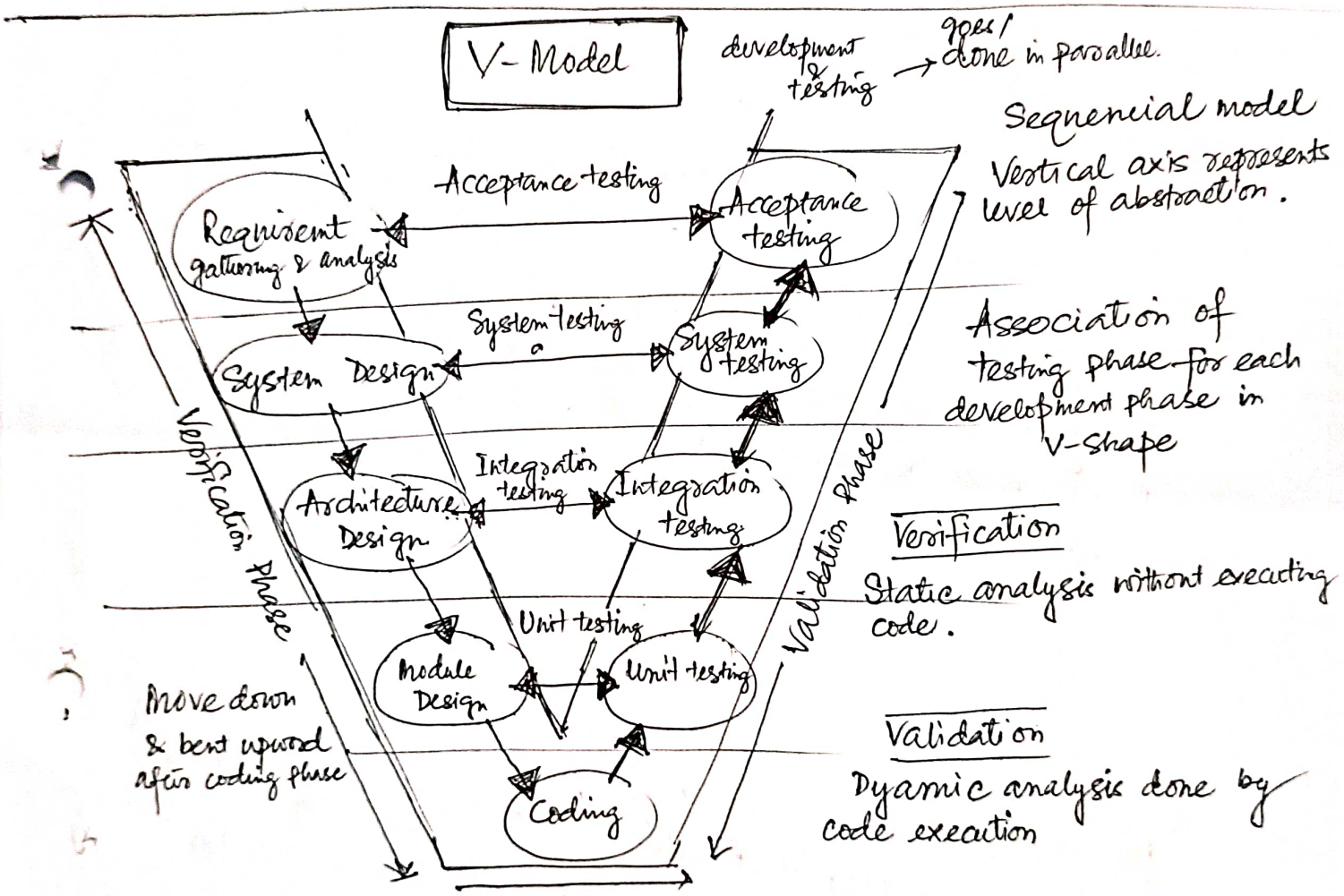
/ preview

Applicable when

Major requirements are defined but the minor details may evolve when time goes.

News technologies are learnt by development team.

Experienced development team.



System design Complete hardware & communication setup.

Architecture design

System design is booken down into modules with different functionalities
— data transfer & communication between internal modules & outside the world.

Module design (Low level design (LLD))

Detail design of modules

Applicable when

Medium-sized ~~Small~~ projects where requirements are clear
Most frequently in medical field

Advantage

Focuses on verification & validation in early life cycle,
—— probability of building error free good quality product

Enables project management to track progress accurately.

Easy to manage due to rigidity of the model

Each phase is deliverable and can be reviewed.

Simple & understandable

Disadvantage

Does not support Iteration — difficult to go back & change functionality.

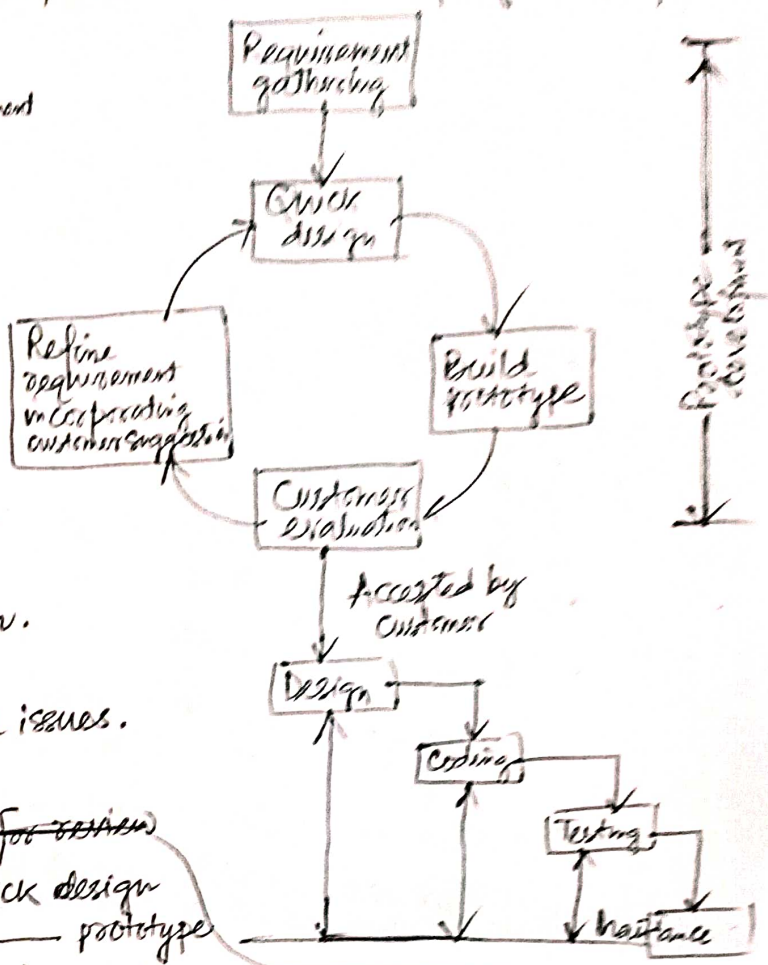
Does not handle concurrent phases / parallel execution

Not good for complex, long project.

PROTOTYPE Model

Prototyping model

Prototype is constructed quickly & serves as basis for system specification



- Before carrying out actual development prototype of a system is built.
- Prototype is like a toy that exhibits limited functionality compared to actual software.
- May contain shortcut implementation (eg. table lookup instead of actual computation)
- Mechanism to understand what customer needs, rather than trying to imagine the working of system.
- Developed prototype can help engineers examine the technical issues.

~~Prototype is submitted to customers for review~~
Requirement is gathered —— quick design is carried out to built prototype —— prototype submitted to customers —— based on customers suggestion ~~proto~~ requirement is ~~updated~~ refined & prototype is updated. —— cycle continues until customers approves prototype.

Experience gathered from prototype involve additional cost, reduces overall cost with unclear customers requirement, minimizes change request from customers (lowering redesign cost)

Types of prototype:

(a) Rapid ~~throwaway~~ throwaway prototype

After customers feedback, the prototype is rejected & new prototype is built from scratch.

(b) Evolutionary prototype

Prototype is incrementally ~~dev~~ refined based on customers feedback

(c) Incremental prototype

Multiple ~~same~~ prototypes are developed individually. These different prototypes are merged into single product.

Applicable when

Requirement is unclear

⑥ Extreme prototype

Full user interface is developed (to draw attention) ~~Very little~~, with limited functionality.

Advantage

- Errors detected at initial stage. (Confusion/ Errors/ Missing functionality)
- Customer statisfaction exists as the customers feels the product at a very early stage.
- Encourages innovation & flexible design.
- Prototype may offer early trainning for fiture user of the software system. → (as Working model of product)
- Increase customer involvement in development.
- Reduces overall time & cost.

Disadvantage

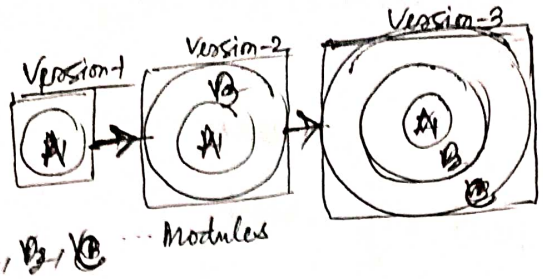
- Too much ~~effot~~ effort:
- Slow & time-taking process — Prototype building & approval
- Cost of developing prototype is total waste if it is thrown-off.
- Prototyping may encourge excessive change request.
- Poor documentation because requirement of customers are changing.
- After seeing prototype customer may think that actual product will be delivered soon.
- Customer may lose interest in final product if he is not happy with initial product.

(Customer may get confused)

- Developer who builds prototype quickly, may end up sub-standard development solution.

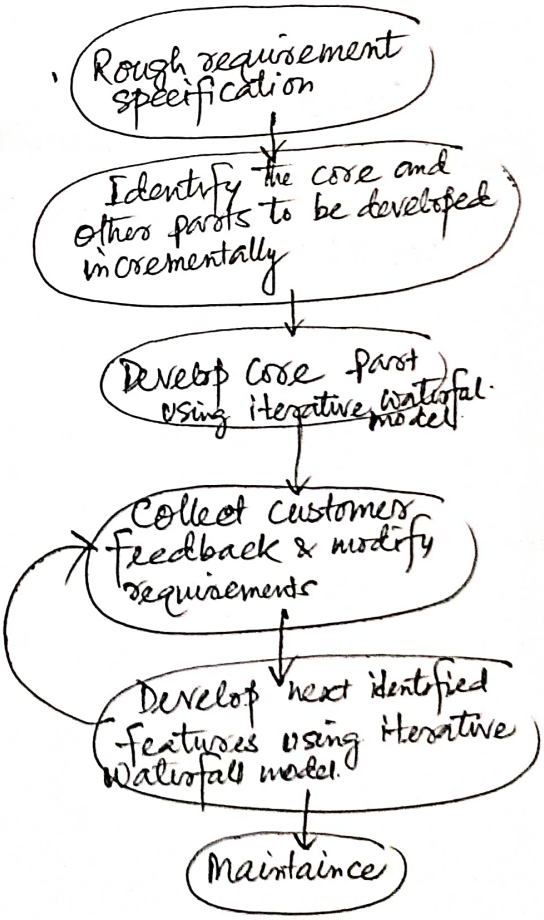
EVOLUTIONARY MODEL

SUCCESSIVE VERSION MODEL
INCREMENTAL MODEL



A, B, C ... Modules

Software is broken down into several modules (or Functional unit)
These modules are constructed incrementally
Initially core module of the system is developed. → Skeleton of the product
Each evolutionary version is developed using iterative waterfall model
Each successive version of the product peforms more useful task than the perrious ones.
User gets a chance to experiment with partially developed software much before the complete version is released.



Applicable when

Very Large product where modules are implemented incrementally.
Customer prefers to receive product in increment
System is partioned into stand alone units
Object oriented software design.

Advantage

Customer's confidence increases as he gets ~~services~~ services/goods constantly from developer.

Reduces error because core modules get tested thoroughly.

Chance to experiment with a partially developed software.

Does not need for large resources at a time.

Good when complete version is impossible due to tight market deadline.

Risk analysis is better.

Supports environmental change.

Disadvantage

Sometimes it is hard to divide problems into several versions that will be acceptable by the customer.

SPIRAL Model

CYCLIC Model
META Model.

Combination of iterative development & linear development
Based on customer's evalution software development enters next Iteration.

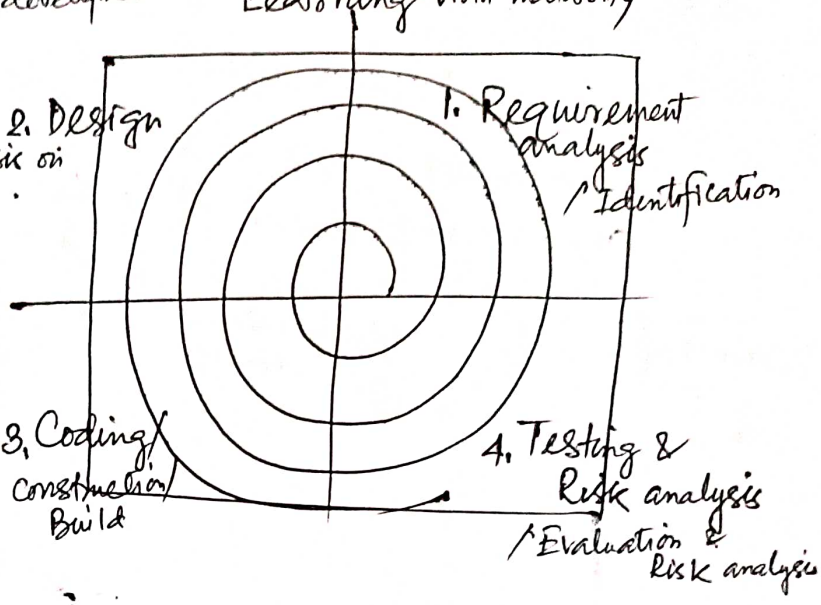
Learning with maturity

Single loop spiral model is waterfall model with high emphasis on risk analysis.

Each phase is represented by sector/quadrat

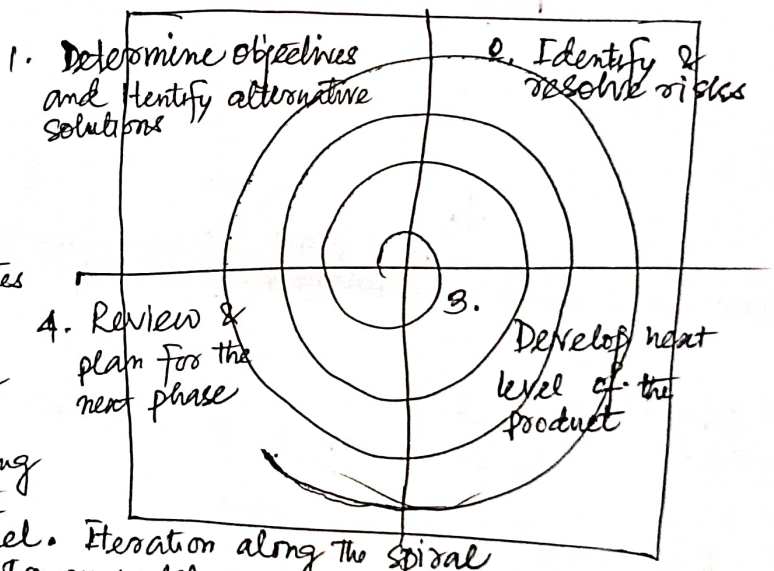
Number of loop depends on developer and project.

Radius of the spiral represents cost whereas angular dimension represents progress.



Why is spiral model called meta model?

It covers all other SDLC models. For example it incorporates stepwise approach like classical waterfall model. ~~loop~~ Existance of loop makes it iterative. Waterfall model. ~~At~~ risk handling stage spiral model uses the approach of prototyping model. Iteration along the spiral can be considered as evolutionary model.



Advantage

Support for better risk handling — useful for mission-critical projects.

Good for large project

Flexibility in requirements / change of requirement is accomodated

Customer satisfaction.

User can see the system easily

Extensive use of prototype

Incremental refinement & incremental release of product around the spiral

Disadvantage

Most complex management among other SDLC model

Expensive — not suitable for small project.

Difficulty in time management as number of loops are not fixed always.

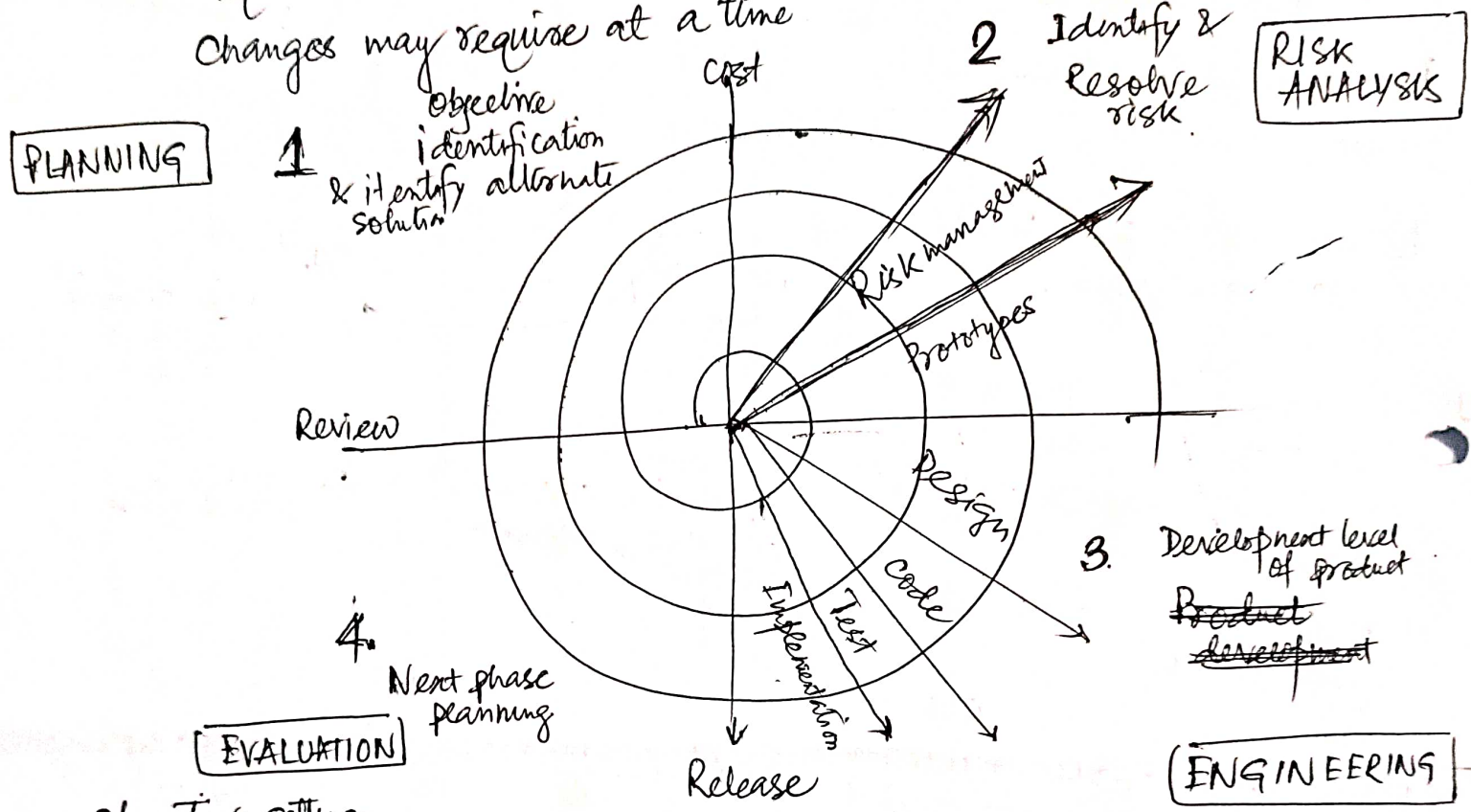
End of the project is not known easily

Spiral may go indefinitely

Excessive documentation required.

Applicable when

Large & high budget
Requirement is unclear & complex
Changes may require at a time



Objective setting

Identification of purpose for that cycle

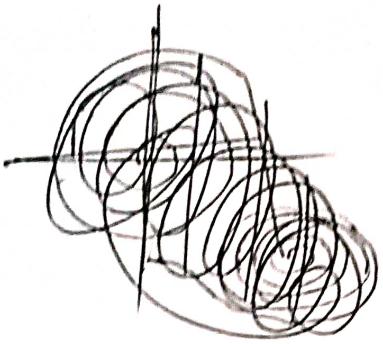
Risk assessment & reduction

Risk pattern

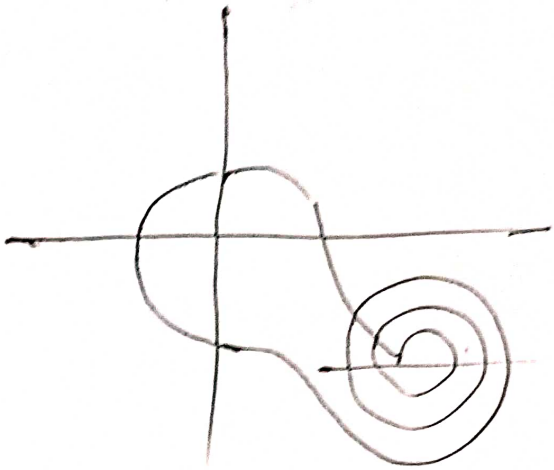
Development & validation

Planning

The project is reviewed —— choice is made whether to continue spiral again.



Spiral in one phase



P- 319, Pressman Book

User interface design process