**Software engineering** is defined as a process of analyzing user requirements and then designing, building, and testing software application which will satisfy those requirements.
It is a systematic, disciplined, cost effective technique for software development.
Software = documentation + program+ instructions
Operating procedures /Instructions =user level, administrative level

**Software myths-> (myth busters)**

- Company with latest tech should not worry about quality
- Software specialist bring things back on track ( it delays the procedure)
- Software is easy to change ( its not)
- Missing components can be ignored and just develop a prototype once(it's a disaster)
- More features , better the software (lol)
- One software is made job is done ( this is jut the beginning lol )
- We cant do testing and cant know its quality
- Only tested codes are deliverable .(is one of the things )
- Develop a working software ( develop that is easy to maintain)

**Product :** that is deliverd to customer.
**Process :** way it is produced

**Measure** : quantitative indication (size, dimensions, efffeciency )
**Measurement :** act of evaluating measure
**Metric** : degree -> planning, organinsng , controlling , improving

**Productivity:** rate of output / time taken

-----------------------------------------------------------------------------------------------------

**Software certification :** though its important , but cant guarantee u skills , as tech changes rapidly
(3 types/area) people , process , product

-----------------------------------------------------------------------------------------------------

**Software development life cycle (SDLC)** :
Planning/requirements-> defining/analysis[srs]-> designing->coding/implementation->testing -> deploy/Maintenance -> [planning]

Design->Dfd :data flow diagram, cohesion and coupling , modules
**Build and fix model:** -

**Waterfall model:**

Feasibility study (availability)

Requirement analysis

Design

Coding and unit testing

Testing integration

Maintenance

Software requirement specif.

| **Advantages** | **disadvantage** |
|---|---|
| Base model | No feedback / no change |
| Simple and easy | We cant know all req in starting<br>No experiment |
| Small projects | No parallelism (can't go back to prev. level) |
|  | High risk |
|  | High maintenance |

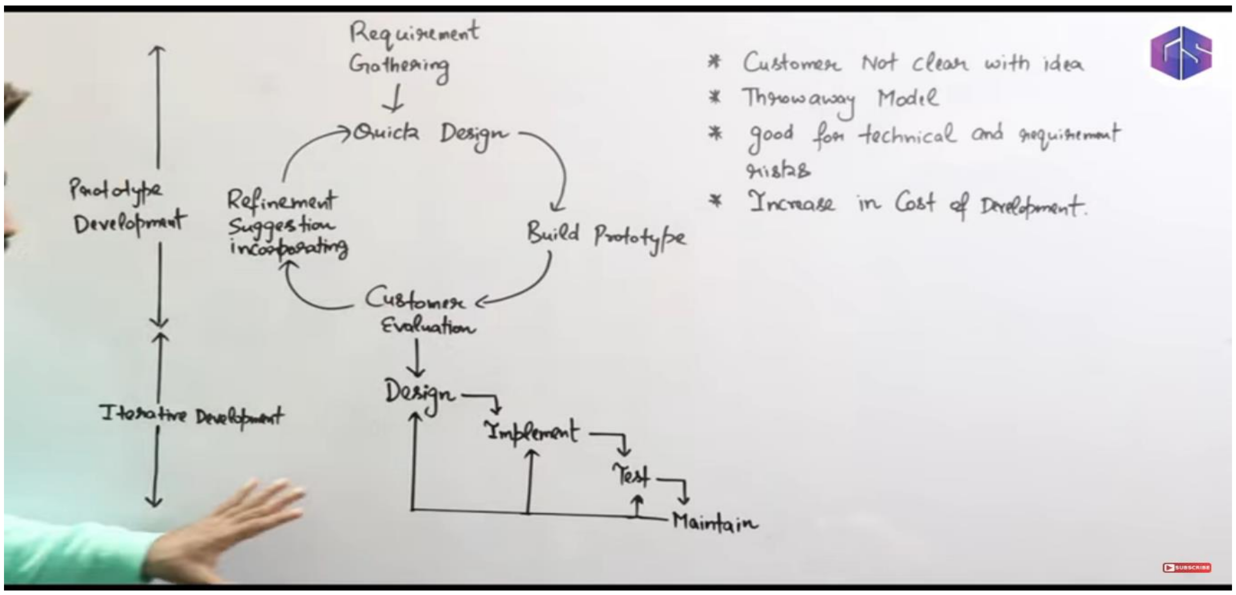
**Iterative model**: ( same as waterfall ,but stages can be repeated <u>major improvement in feedback</u> )

|  |  |
|---|---|
| Base momdel |  |
| Simple and easy | No pallelism |
| Small project | Rigid (no changes in feasibility study) |
| feedback | No intermediate delivery (less testing) |

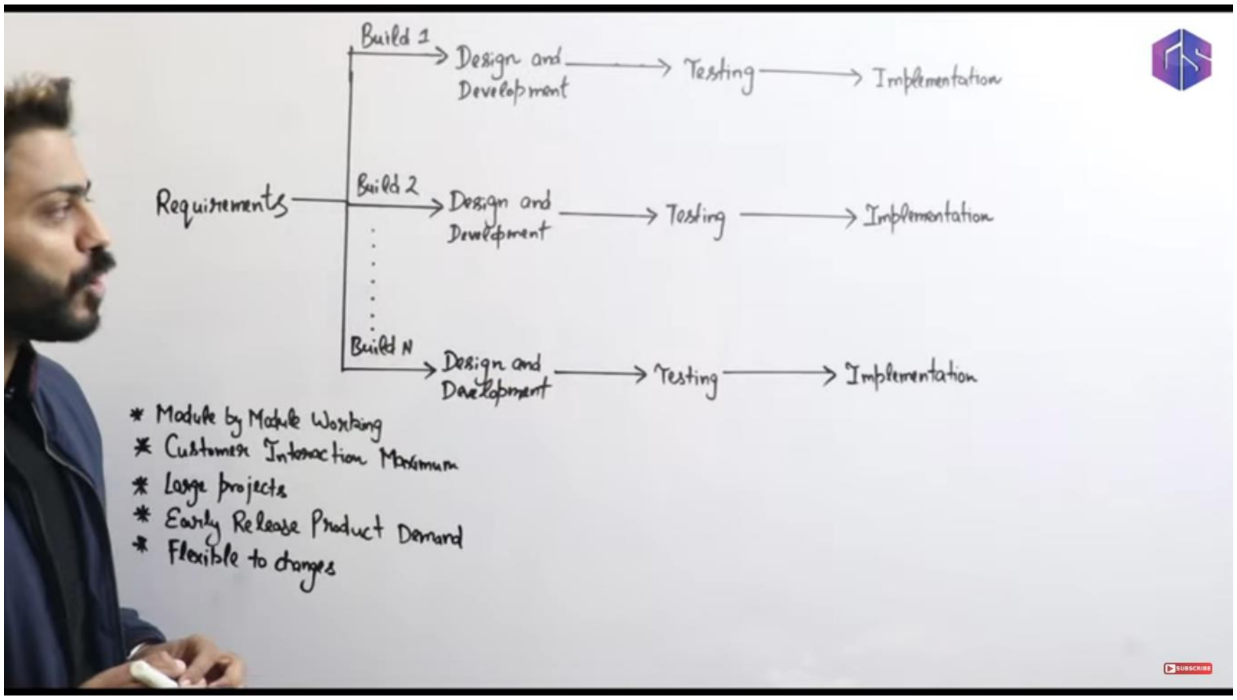
**Rapid application development :** user participation is essential, evaluation, feedback . less planning , more development and come up with a prototype

- Reusable components
- Skilled developers Parellelism resource sharing

**Prototype modeling (dummy model )**

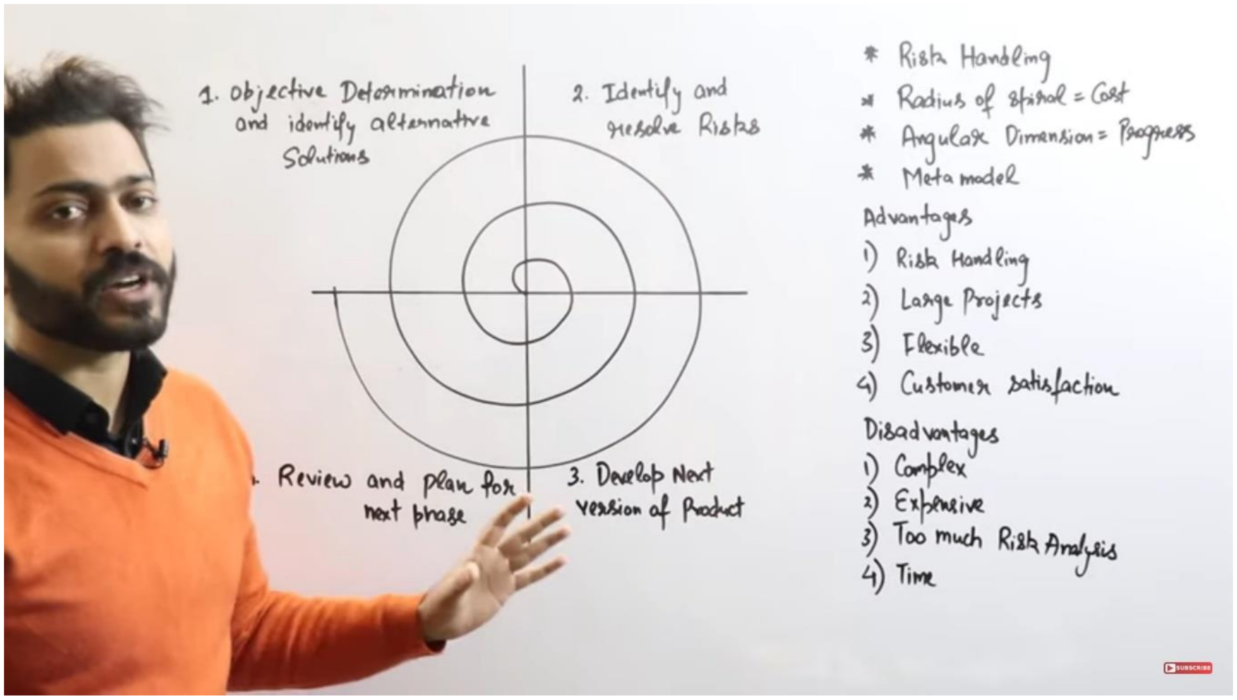


**Evolutionary momdel**: same as above , but user can make changes in his req
**Incremental modeling :** (module by module development)



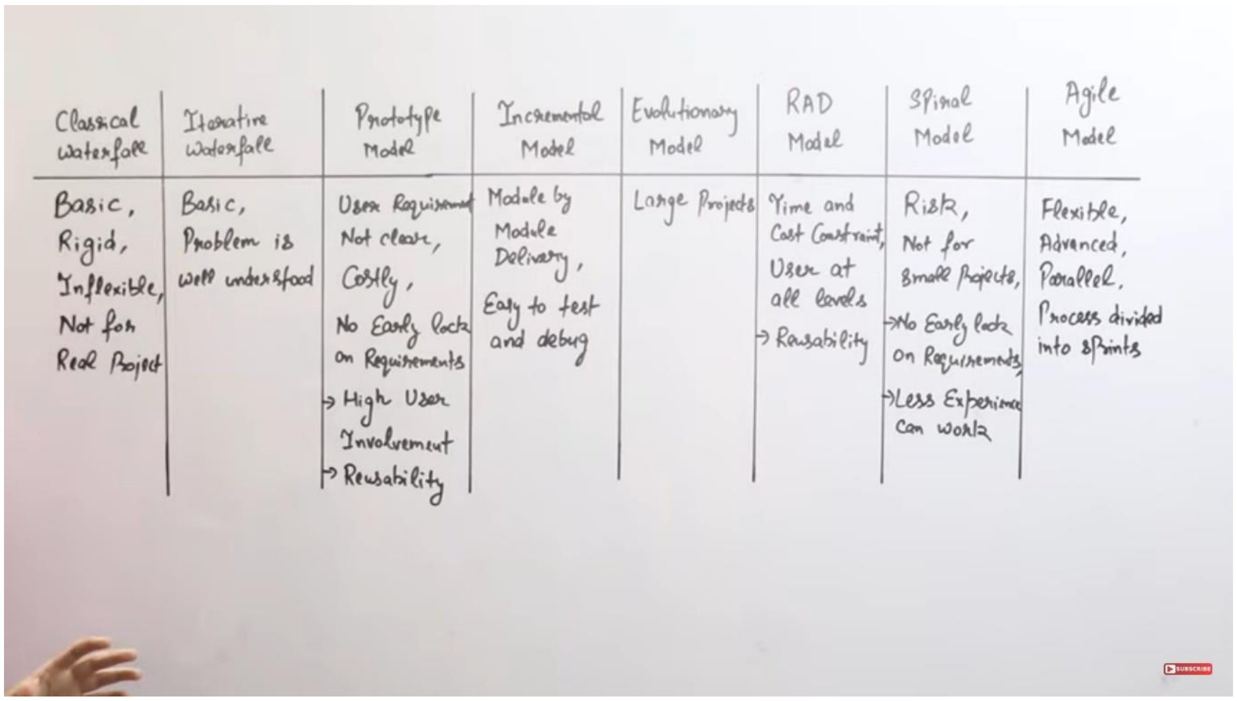
Early relaease , large prjectss into subprojects
Example : ims -> teachers , students , alumni
 Mobile apps upgrade etc

**Spiral model ( risk handling in every phase ) meta model**


1. Objective Determination and identify alternative Solutions
2. Identify and resolve Risks
3. Develop Next version of Product
. Review and plan for next phase
* Risk Handling
* Radius of spiral = Cost
* Angular Dimension = Progress
* Meta model
Advantages
1) Risk Handling
2) Large Projects
3) Flexible
4) Customer satisfaction
Disadvantages
1) Complex
2) Expensive
3) Too much Risk Analysis
4) Time

1-planning 2-risk analysis 3-development 4-assessment
**Disadv :** constraints changes, expertise

**All in one :**



| Classical waterfall | Iterative waterfall | Prototype Model | Incremental Model | Evolutionary Model | RAD Model | Spiral Model | Agile Model |
|---|---|---|---|---|---|---|---|
| Basic, Rigid, Inflexible, Not for Real Project | Basic, Problem is well understood | User Requirement Not clear, Costly, No Early lock on Requirements → High User Involvement → Reusability | Module by Module Delivery, Easy to test and debug | Large Projects | Time and Cost Constraint, User at all levels → Reusability | Risk, Not for Small Projects, → No Early lock on Requirements, → Less Experience Can work | Flexible, Advanced, Parallel, Process divided into sprints |

Certification is process of formal confirmation of various characterstics