- SCS1207 -

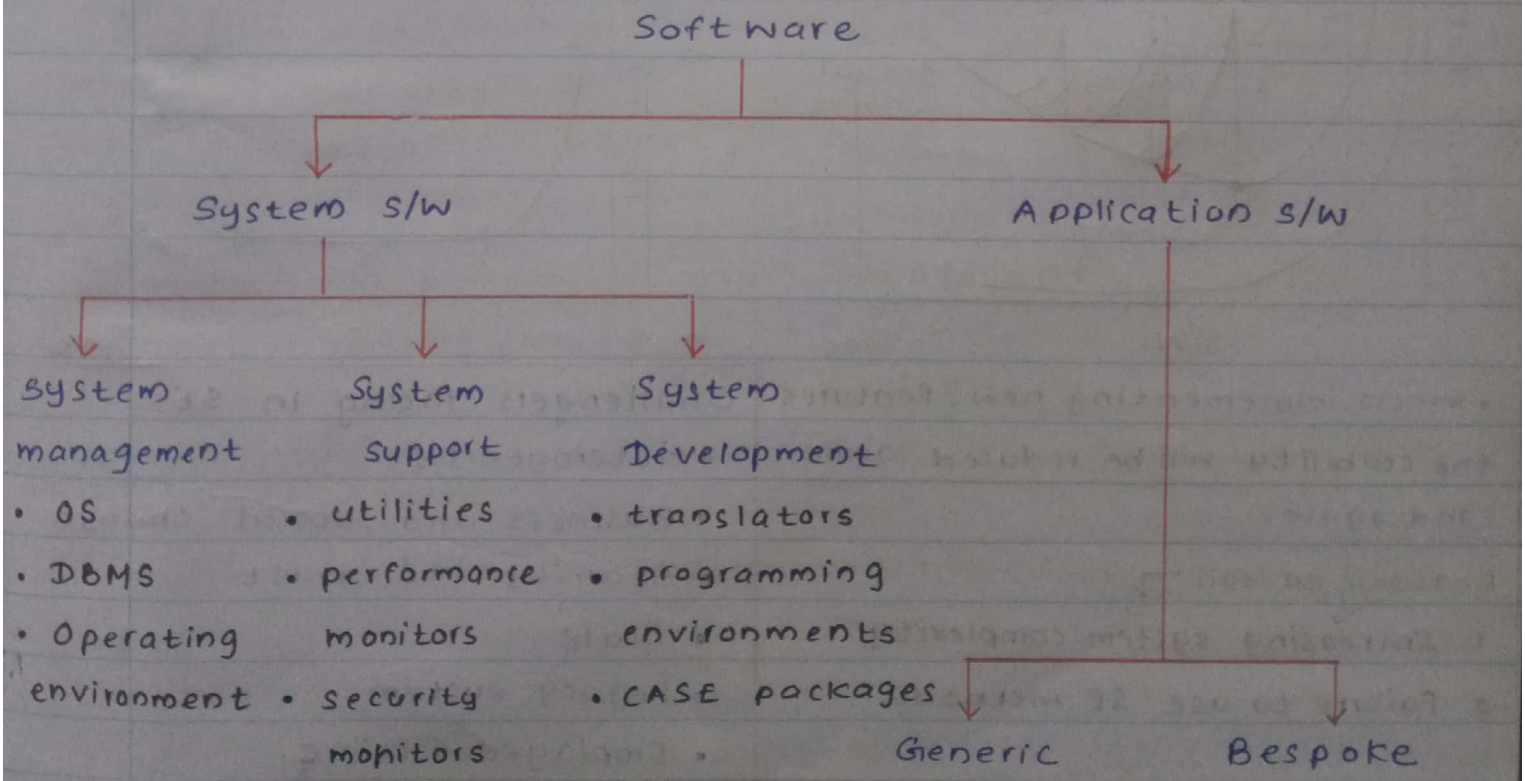
1. Introduction to Software Engineering

- A **software** is a set of **machine instructions** to the computer processor to perform **specific operations**.

- A **software system** is a collection of **inter communicating** components, **programs**, **configuration files**, **system documentation** and **user documentation**.

s/w projects vs other projects.

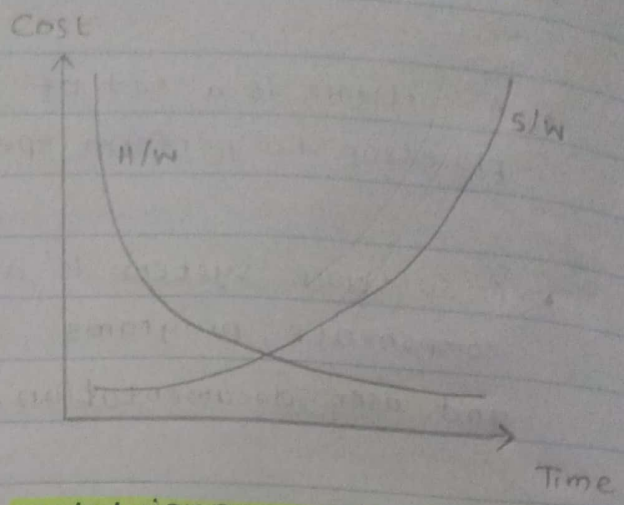
- difficult for the customer to specify requirements.
- requirements change regularly.
- s/w is intangible
- customer and developer has communication gaps.
- difficult to test exhaustively.



Application S/W types.

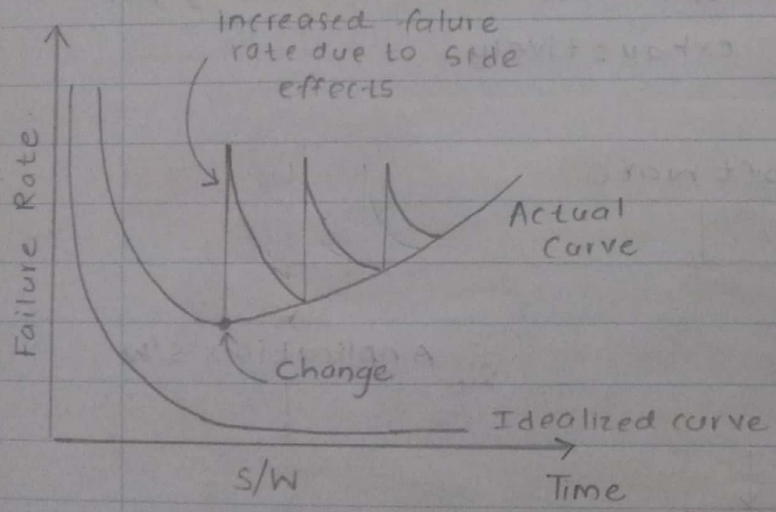
- stand alone
- interactive transaction based
- embedded control
- Batch processing
- Entertainment
- Modelling and stimulation
- Data collection
- System of systems.

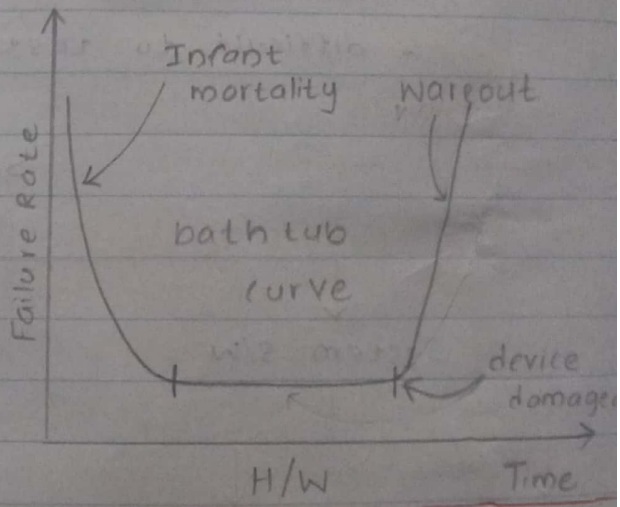
Cost of H/W vs S/W



- maintainance cost is higher than development cost

Failure curves for S/W and H/W





- When implementing new features the stability will be reduced again and again.

Reasons to fail ↓

1. Increasing system complexity
2. Failure to use SE methods.

Challengers facing in SE

- Heterogenity
- Business and social change
- Security and trust
- Scale
- Legacy system
- Employee forklore.

Why we need to have a deciplinary way in SE?

- High failure rate
- Life critical / mission critical
- High impact with GNP
- more complex
- most systems are S/W controlled.

Features of a good S/W

- Maintainability
- Dependability
- Security
- Efficiency
- Acceptability

*

- Cost effective
- User friendly
- Userbility
- Interopability
- Flexibility.

- **SOFTWARE ENGINEERING** is an engineering discipline that is concerned with all aspects of s/w production.

Activities of S/W process.

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

- SOFTWARE PROCESS is a systematic approach or structured set of activities to build up a software.

→ plan driven process
- all activities are planned and progress is measured against this plan.

→ agile processes.
- planning is incremental and easier to change the process.

Ethics in SE

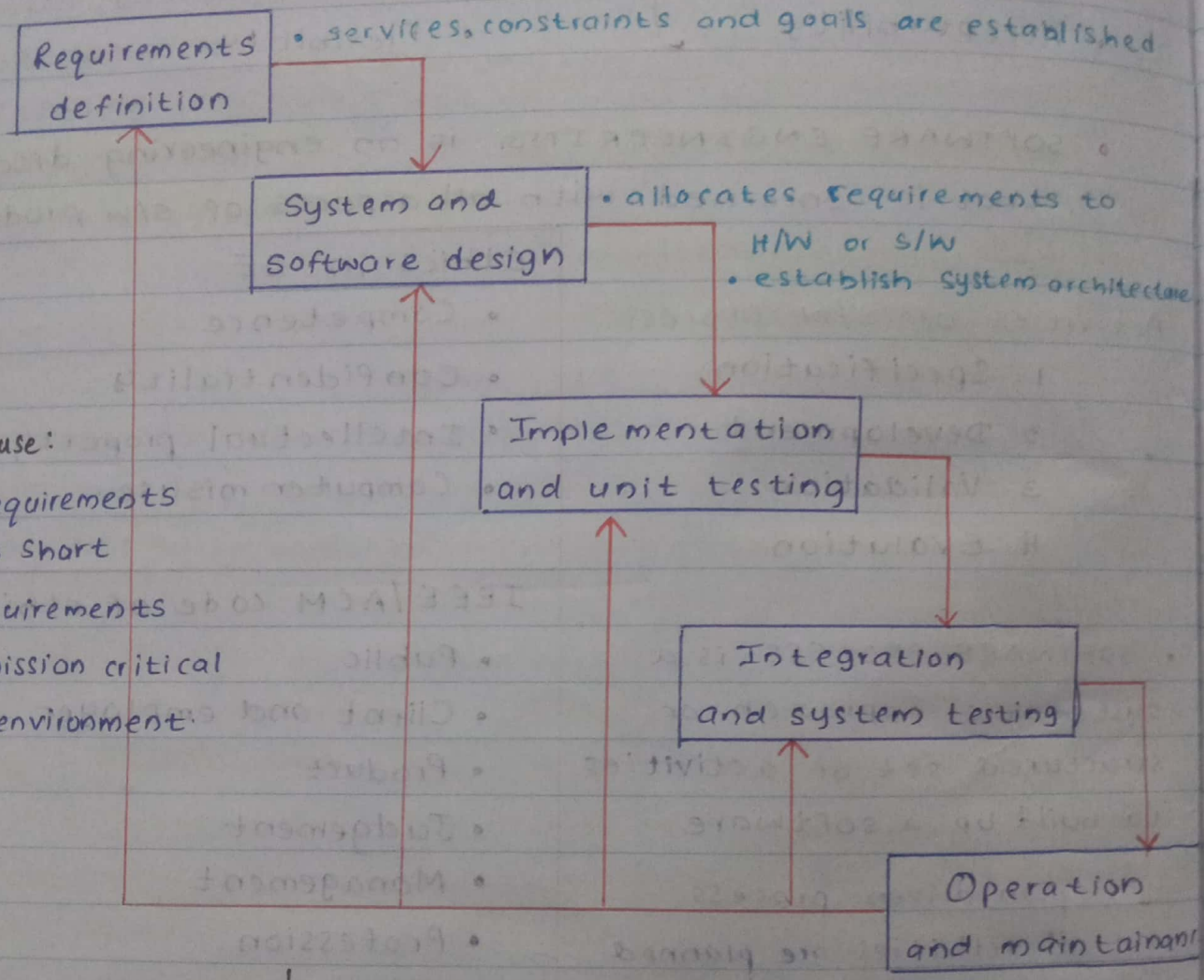
- Competence
- Confidentiality
- Intellectual property rights
- Computer misuse.

IEEE/ACM code of ethics.

- Public
- Client and employer
- Product
- Judgement
- Management
- Profession
- Collegues
- Self

WATERFALL MODEL

- plan driven
- no overlapping
- linear sequential
- document driven
- clear and stable requirements
- not allowed to come back.



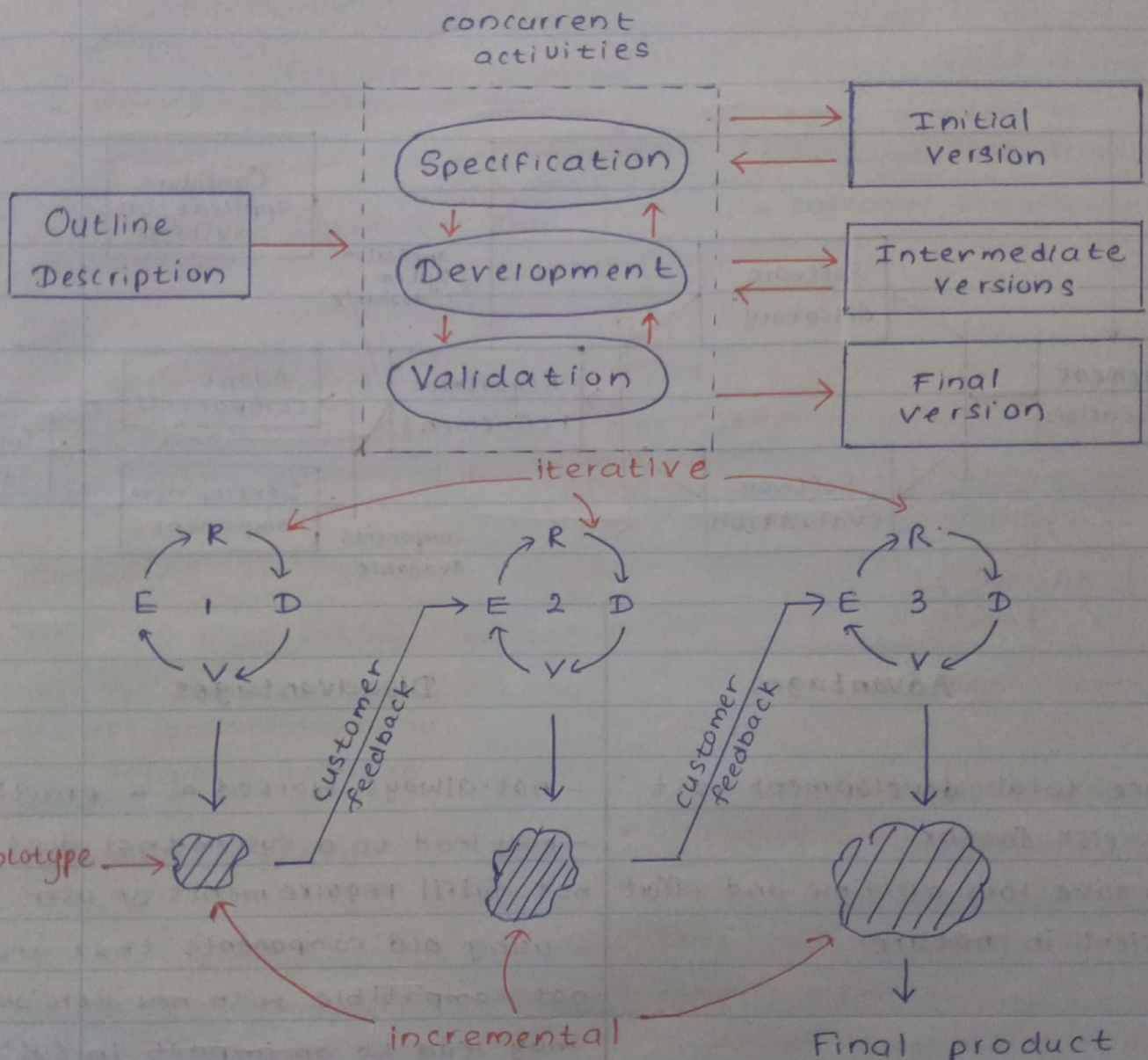
When to use:
- stable requirements
- project is short
- clear requirements
- life or mission critical
- stable environment

Advantages	Disadvantages
- No overlapping - Small project with stable & clear req. - Quality assured in every phase - Elaborate documentation	- Errors can be fixed only during proper phase - Not suitable for complex projects. - Testing comes at end - Minimal customer involvement

Smaller and mid sized projects with clearly defined requirements

INCREMENTAL DEVELOPMENT

- initial implementation, getting feedback from users and evolveing the software through several versions.
- broken down to mini projects
- highest priority requirements are tackled first



Advantages

- requirement implementation cost is reduced
- easier to get customer feedback
- critical requirements can be issued in early stage
- easy to identify errors

Problems

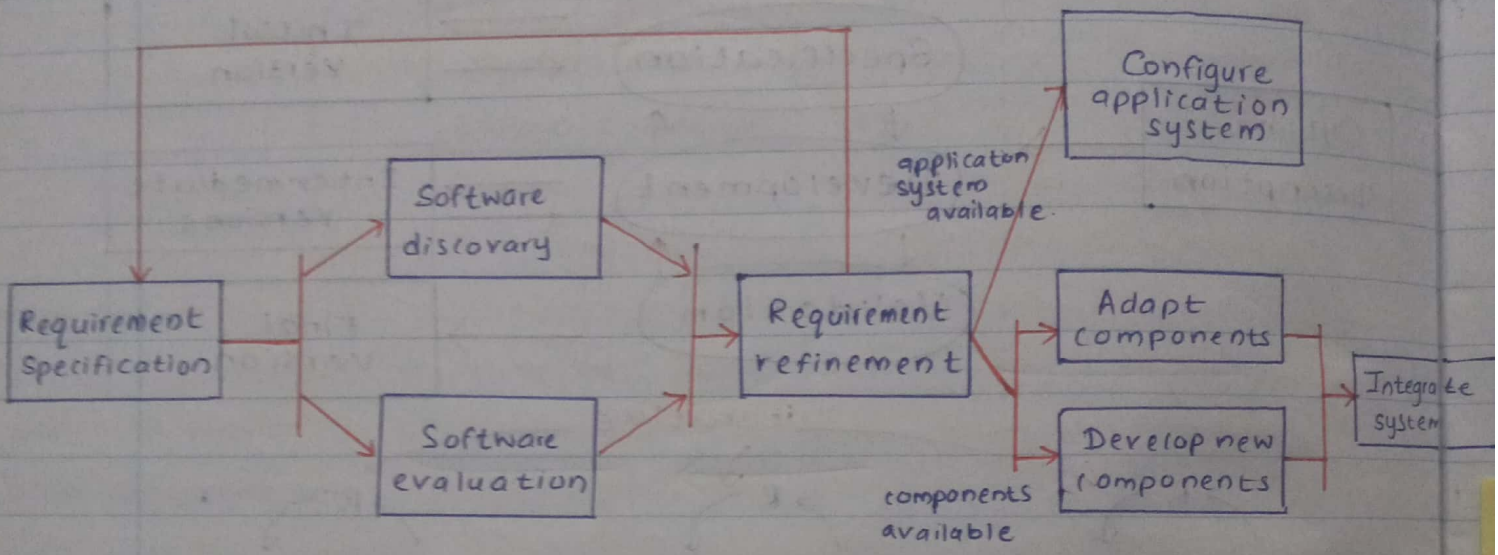
- Process is not visible
- System structure tends to degrade as new increments are added.

Projects with loosely coupled parts and with complete, clear requirements.

REUSE ORIENTED SOFTWARE ENGINEERING

(integration and configuration)

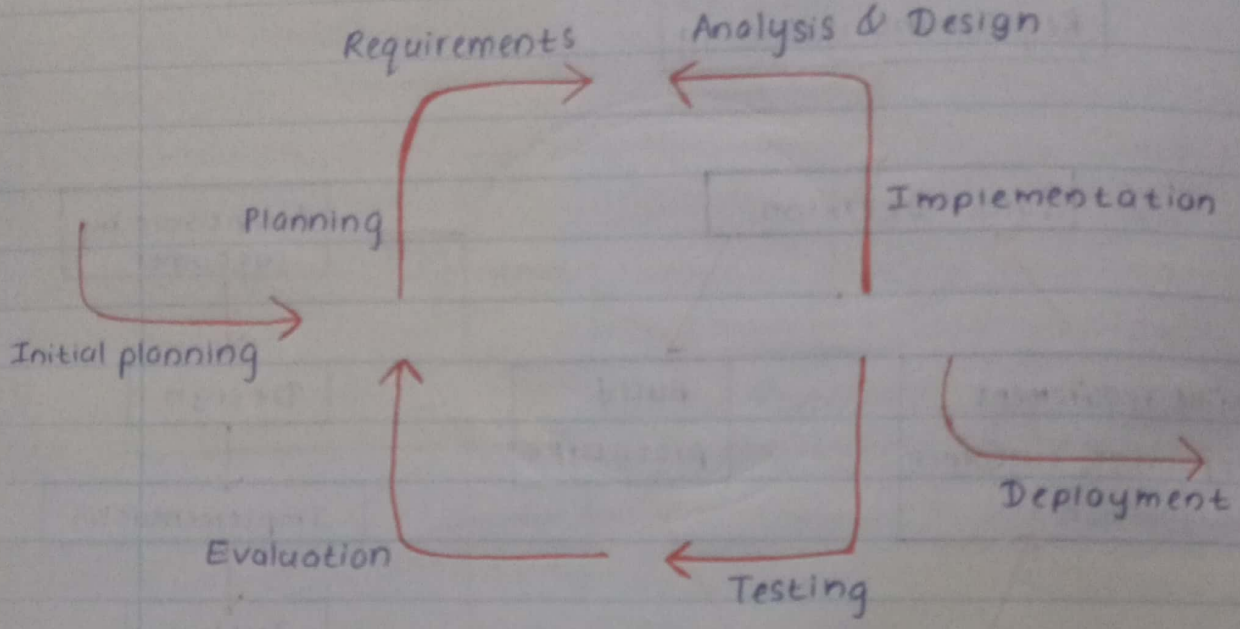
- method of s/w development in which a program is refined by producing a sequence of prototypes called models.



Advantages	Disadvantages
- reduce total development cost - low risk factor - can save lots of time and effort - efficient in nature	- not always worked as a practise - may lead to a system that does not fulfil requirements of user - using old components that are not compatible with new versions may lead to an impact in system evolution.

ITERATIVE DEVELOPMENT MODEL

- begins with smaller set of requirements.
- those req are used to analyze and gradually implement features.
- then those features are evaluated and tested to identify new req



Advantages

- coding begins early
- cost effective
- streamlined management
- easy to spot bugs and defects

Disadvantages

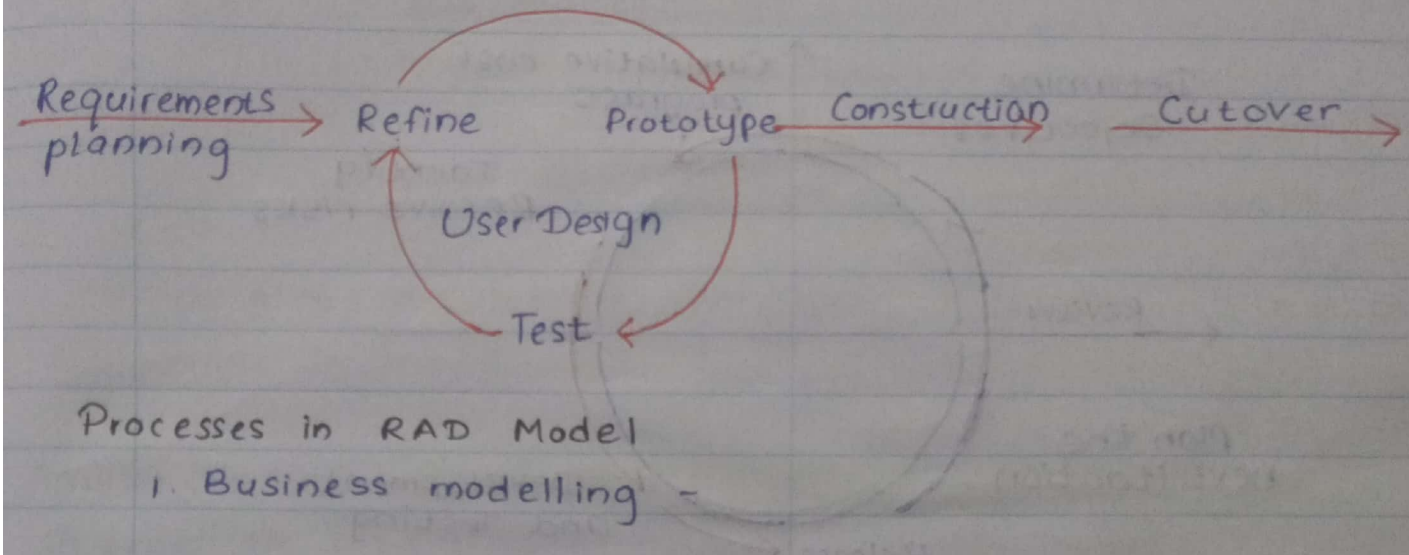
- difficult to analyze risks
- potential design and architecture issues in the later phases
- too reliant on baseline plan
- Resource-heavy model

Large scale products with multiple modules

Clearly defined requirements

RAPID APPLICATION DEVELOPMENT (RAD)

- iterative
- extremely short development cycle.
- requirements should be well understood.



Processes in RAD Model

1. Business modelling -
2. Data modelling -
3. Process modelling -
4. Application generation -
5. Testing and turnover -

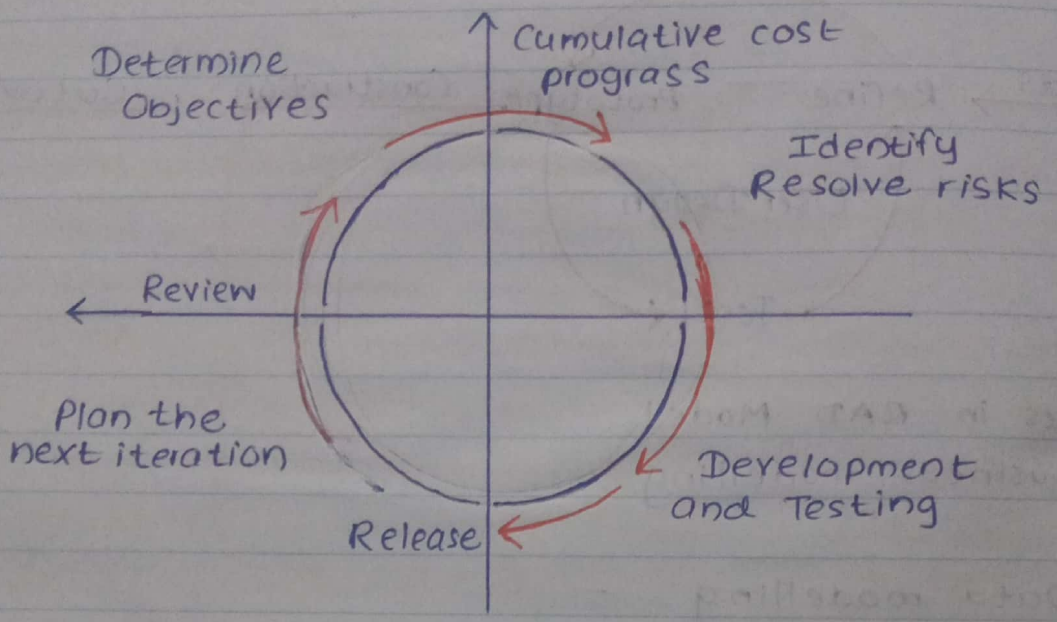
Issues in RAD

- sufficient human resources
- developers and customers should be committed to project
- will be problematic if the system cannot be modularized
- not applicable when technical risks are high

Systems that need to be produced in short time and have known requirements.

SPIRAL MODEL

- combines waterfall and iterative with risk assesment.
- delivers projects in loops.
- each phase addresses whatever problem has the greatest risk



Advantages

- Precisely documented
- Accurate time and budget estimates
- Excellent risk assesment
- Can apply changes to new iterations

Disadvantages

- Success depends on skilled risk manages
- Large resource pool
- Time consuming.

Larger projects with complex requirements
New projects with multiple testing stages.