Software Engineering & Project Management

Unit - 1

Software Engineering involves 3 basic steps :-

1) Designing
2) Implementation & Deployment
3) Maintainance.

After applying these 3 steps, we get Software Product.

Software Development Life Cycle (SDLC) :-

It is a process used by software industry to design, develop and test high quality software.
It involves 5-6 stages.

Following are the stage of SDLC :-

① Requirement Analysis :- It is the most important and necessary stage in SDLC. This stage deals with the requirement of resource needed to build the software.

There are 4 types of requirements.

(a) Software Requirement :- Software needed to build the required product.

(b) Hardware requirement :- Hardware which will be used to make the product.

(c) Functional requirement :- those which are needed for working of software.

(d) Non-functional requirement :- feasiblity / reliability

(2) Design :- This phase deals with the designing part which means how exactly the software will look to the consumer.

(3) Coding :- In this phase, actual development begins and the programming is built. Developer have to follow the coding guidelines described by their management and programming tools.

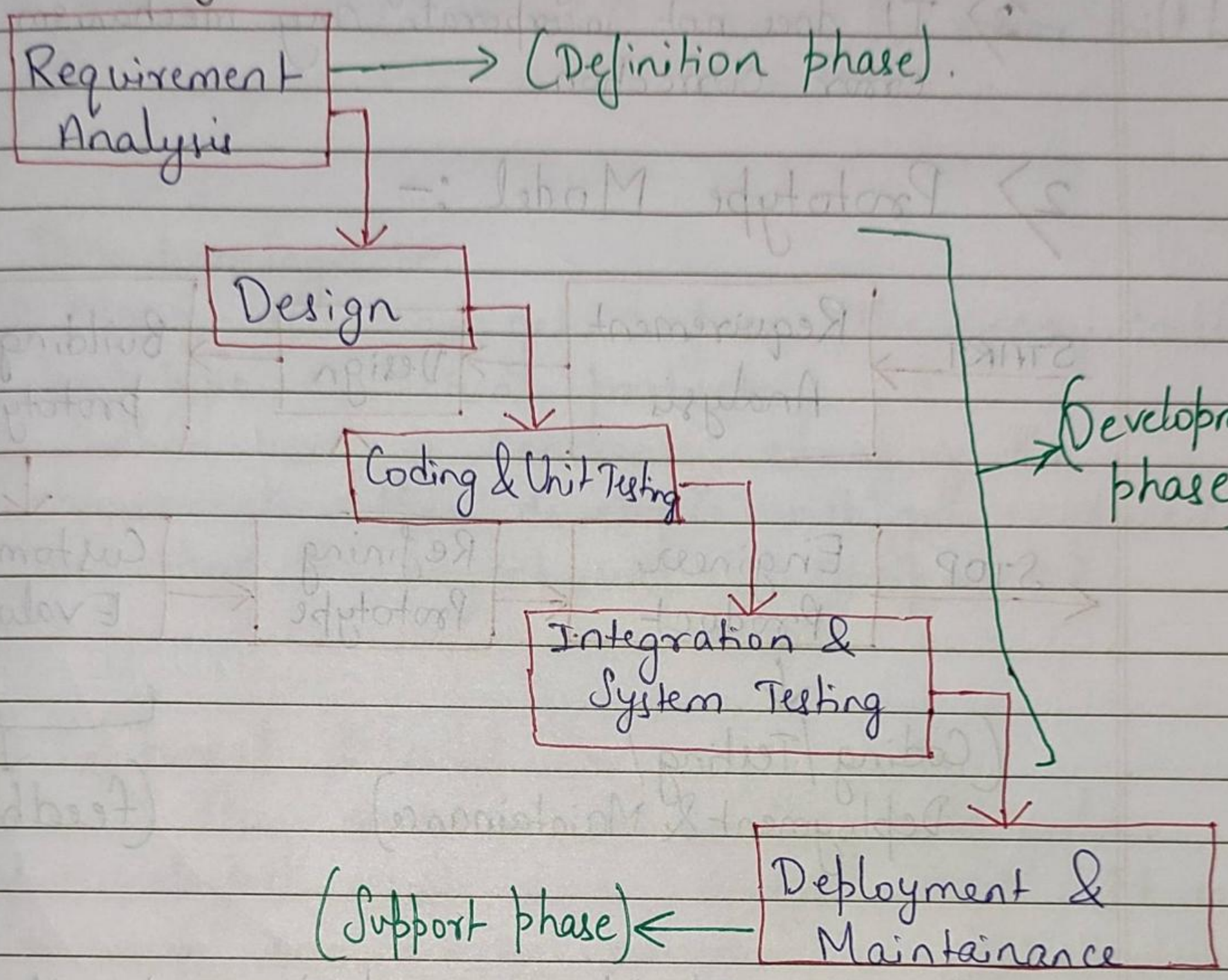
(4) Testing :- After the code is generated, it is tested against the requirement to make sure that the product are solving the needs addressed and gathered during requirement stage.

(5) Deployment & ~~Emple~~ Maintainence :-
Once Software is certified and no bugs or errors are stated, then it is deployed. After the software is deployed then its ~~mainen~~ maintainance begins.

Example of Some common SDLC :-

(a) Waterfall Model
(b) V-Model
(c) Prototype Model
(d) Spiral Model
(e) Agile Model
(f) XP
(g) Scrum
(f) RAD.

① Waterfall Model :-



This is a very simple and basic model of SDLC. It works just like a waterfall. Like Waterfall is which water that drops down can't go upwards. In this model too, we can't go back to previous stage after reaching next stage. If any error occurs then we have to restart from the 1st stage.

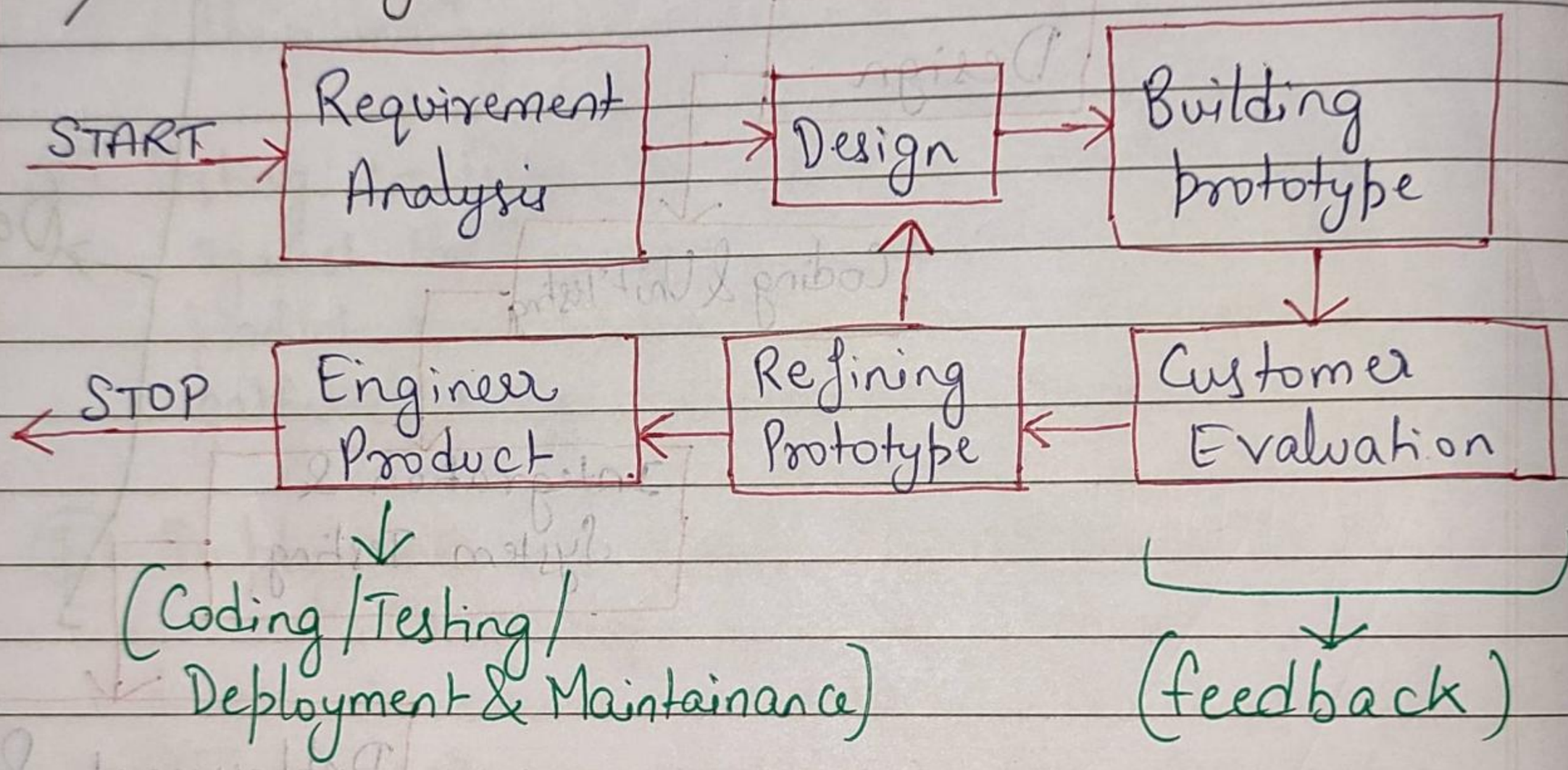
Advantages :-

1) This model is very simple and easy to understand.
2) Works well for small project and projects where requirements are well understood.
3) It is usually not time taking model.

Disadvantage :-

1) Can't be used for large & complex project.
2) It does not incorporate any mechanism for error correction.

2) Prototype Model :-



In this Model, we don't make the entire product initially. We just make a prototype and show that prototype to the customer and if customer ask for any changes we will again redefine the prototype and this loop will run untill customer gets satisfied. On customer verify the prototype, that prototype is selected and the actual product is made with the help of this prototype.

Date
Page

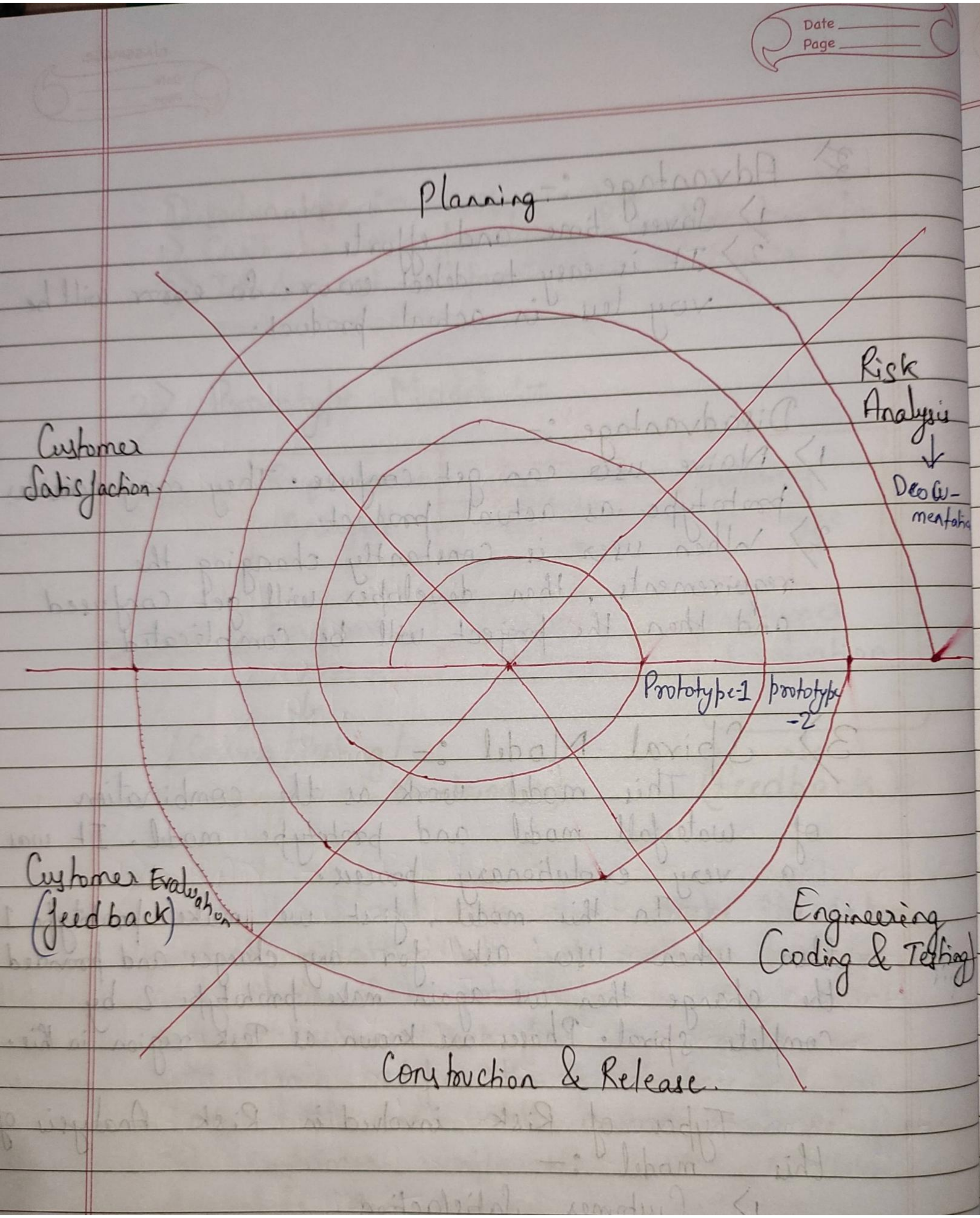
3) Advantage :-
1) Saves time and efforts
2) It is easy to detect error. So error will be very less in actual product.

Disadvantage :-
1) Naive user can get confuse. They may consider prototype as actual product.
2) When user is constantly changing the requirements, then developer will get confused and then the project will be complicated.

3) Spiral Model :-
This model work as the combination of waterfall model and prototype model. It was a very evolutionary process.

In this model, first we make prototype 1 then when user ask for any changes and provided the change then we again make prototype 2 by complete spiral. Phases are known as Task region in this.

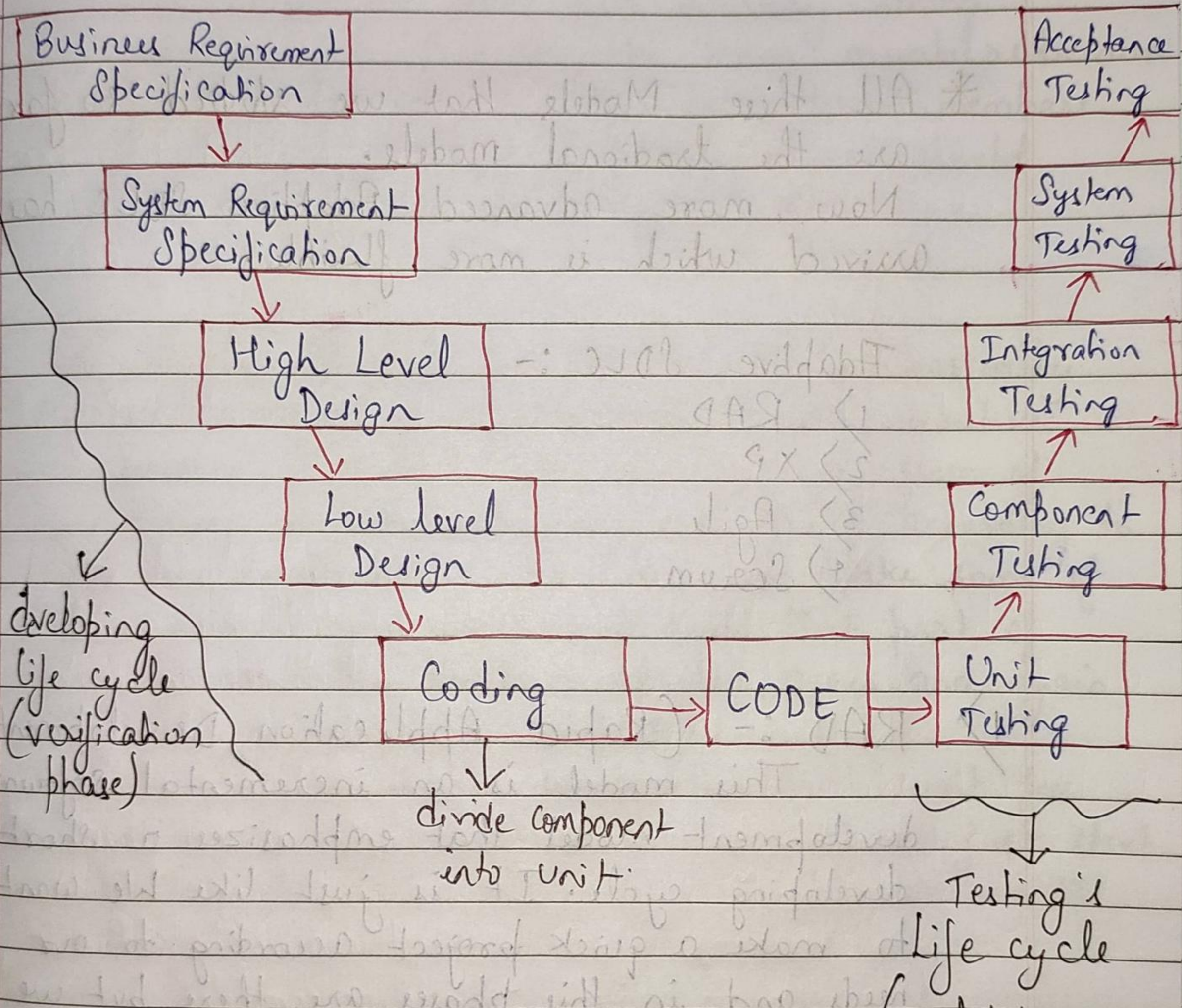
Types of Risk involved in Risk Analysis of this model :-
1) Customer Satisfaction
2) Cost
3) Technological failure
4) Time to market.



(4) V-Lifecycle Model (V-Model) :-

↓

Verification / Validation.



Business Requirement :-

1) Time
2) Cost
3) Quality
4) Time to market

This model is almost same as waterfall model. The difference is that in V-Model testing will be done parallely.

* All these Models that we studied so far are the tradional models.
Now, more advanced Adaptive SDLC has arrived which is more flexible.

Adaptive SDLC :-
1) RAD
2) XP
3) Agile
4) Scrum

1) RAD :- (Rapid Application Development) :-
This model is an incremental software development model that emphasizes a short developing cycle. It is just like we want to make a quick project according to our needs and in this phases are there but we can choose which phase we want to complete first.

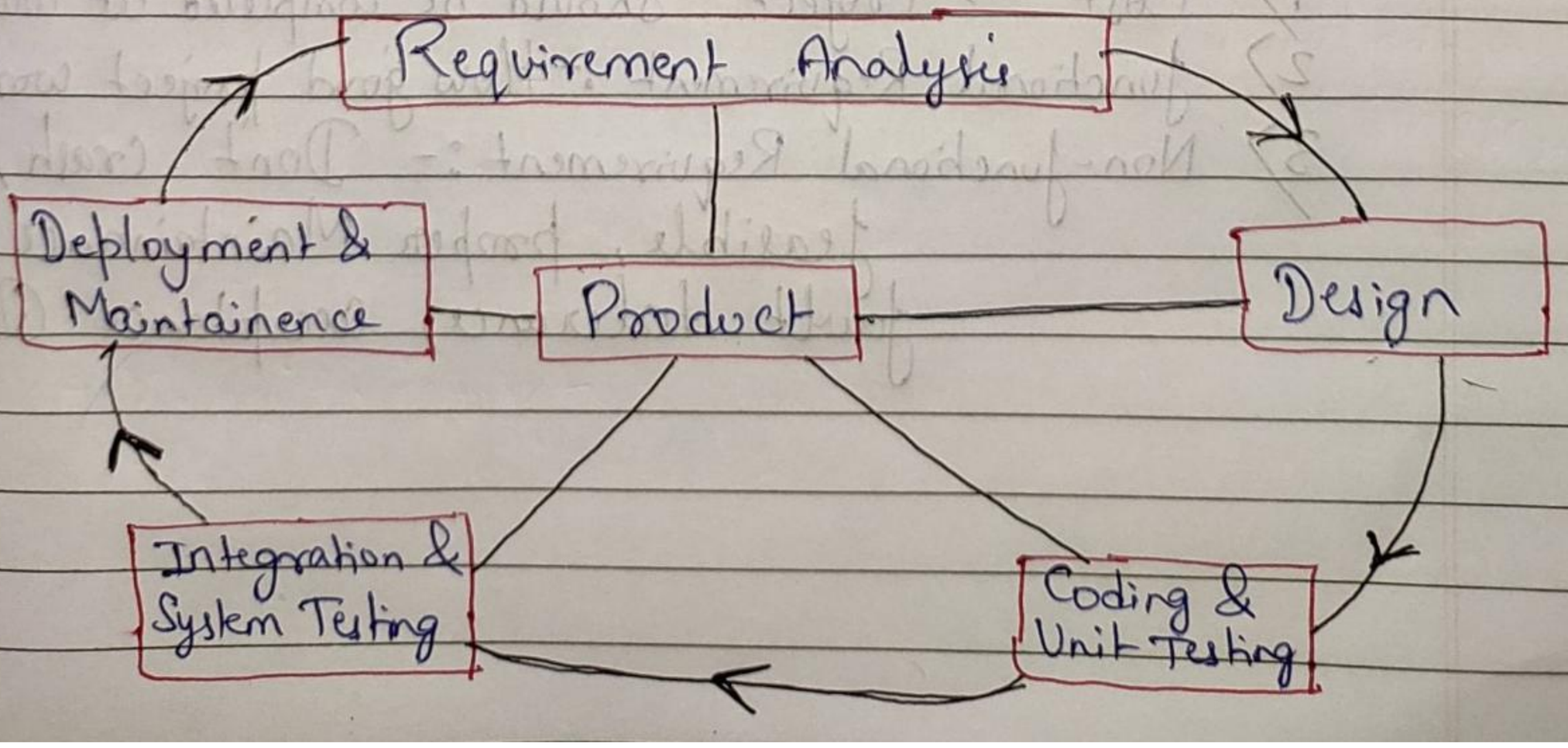
Phases :-
1) Business Modeling
2) Data Modeling
3) Process Modeling.
4) Application Modeling
5) Testing and Turnover.

Advantages :- 1) Less time to develop.
2) ~~Reuse~~ Reusable component

Disadvantages :- We need a very strong communication between the team members because different phases may be done by different members so to make a proper project everyone should know what other's are doing at that time.

2) Agile Model :- This model is basically a combination of incremental and iterative process. In this we can arrange flow of phases in whatever way we want according to the requirement. In this, we take some of customer requirement and build that part of software. The built is repeated again and again until development completed.

In simple words, we don't wait for requirement analysis phase to complete and just start building project with whatever data we received till that point.



Advantage :- 1) Less time
2) Less Resource
3) Less effort

Disadvantage :- 1) Requirement is not fixed. So there is no clear idea of what to make.
2) It totally depend on customer interaction. Efforts will be wasted if customer changes its mind.

* It can't be used for Complex Project.

3) XP (Extreme Programming) Model :-
The main concept of this model is that there is no concept at all. We can make the project in whatever way we like.
Customer Satisfaction is the key part of this model. There is no phases at all.

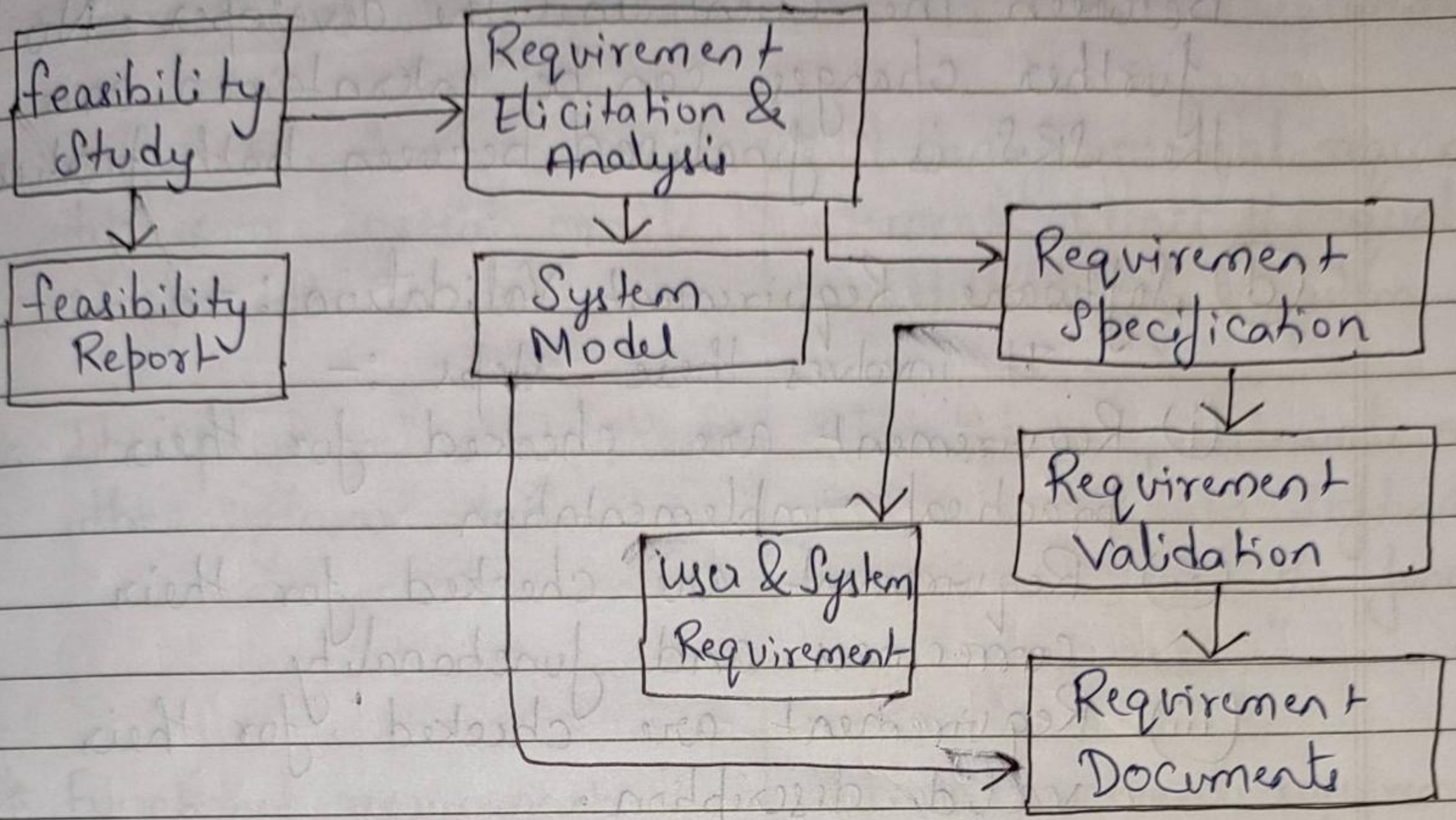
Factors affecting customer satisfaction :-
1) Cost → Project should be completed at low rate.
2) Functional Requirement :- How good project works.
3) Non-Functional Requirement :- Don't crash, Reliable, feasible, proper Maintainance, fault tolerance, acceptance Quality, etc.

Effect of designing a prototype on overall cost of Software project :-

Prototyping may have some initial cost of developing, but it reduces the overall budget by helping your product to be free of errors or glitches. Furthermore, prototyping also helps to understand intrinsic flaws that can be improved during the product development process. So, the effect of designing a prototype on overall cost of Software product is to actually reduce the additional cost of restructuring and reframing it after its full-fledged development.

4. Write short note on Requirement Engineering.

Ans→ Requirement Engineering is the process of defining, documenting and maintaining the requirements. It is the process of gathering and defining service provided by System.



It involves following steps :-

(a) Feasibility study :- In this we check if making a project is feasible or not on these parameters :-

(i) Technical
(ii) Operational
(iii) Economic

(b) Requirement Elicitation & Analysis :-

* Getting only right person involved.
* Often the client do not know what they want
* Clients may have conflicting requirements.

(c) Requirement Specification :- It is an official document which is considered as an agreement

between the client and the developer. No further changes can be entertained once the SRS is finalised between both parties.

(d) Software Requirement Validation :-

It involves these steps :-

(i) Requirement are checked for their practical implementation

(ii) Requirement are checked for their correctness and functionality.

(iii) Requirement are checked for their valid description.

(iv) It is checked whether or not requirements are complete.

5. Discuss the difference between functional and Non-functional Requirements.

Ans→ functional Requirement :- These are requirement that the end user specifically demands as basic facilities that the system should offer.

Non-functional Requirement :- These are basically the qualities constraints that the system must satisfy according to the project contract.

Functional Requirement	Non-functional Requirement
* A functional requirement defines a system or its component	* A non-functional requirement defines the quality attribute of a system.
* It specifies "What should the software do?"	* It places constraints on "How should software system fulfil the functional requirements."
* functional requirement is specified by user.	* Non-functional requirement are specified by technical peoples.
* It is captured in use case.	* It is not captured in use case.
* Helps to verify functionality of software	* Helps you to verify performance of software.

Software Project effort & Cost Estimation :-

- Effort Estimation :- Effort estimation is the process of forecasting how many effort is required to develop or maintain a software application.

It is traditionally measured in hour worked by a person, or money needed to pay for this work.

Cost Estimation :- For a new software project it is necessary to know how much it will cost to develop and how much development time will it take.

Uses of cost estimation :-

1. During the planning stage, one needs to choose how many engineers are required for the project and to develop a schedule.

2. In monitoring the project's progress, one need to acess whether the project is progressing according to the procedure and take corrective action, if necessary.

Cost Estimation Model.

- Static Single Variable Model
- Static Multivariable Model.

(a) Single-static Variable Model :- When a model make use of single variable to calculate the desired values such as cost, time, efforts, etc then it is said to be single-static variable model.

$$C = aL^b$$

Where. $C \rightarrow$ Cost
a & $b \rightarrow$ Constant
$L \rightarrow$ Size (Lines of Code).

Example :-
The Software Engineering Laboratory (SEL) established a model called SEL model for estimating its software production.

$$E = 1.4 L^{0.93}$$
$$DOC = 30.4 L^{0.90}$$
$$D = 4.6 L^{0.26}$$

E → Efforts (Person per Month)
DOC → Documentation (No. of Page)
D → Duration of development (Months).
L → No. of lines of per code

(b) Static Multivariable Model :- They use multiple variables describing variou aspect of the software development environment.

Example :-

Walton & Felix Model :- This model is developed by IBM.

$$E = 5.2 L^{0.91}$$

$$D = 4.1 L^{0.36}$$

$$I = \sum_{i=1}^{29} w_i k_j$$

↓

E → Effort

D → Duration of Development

I → Productivity index.

Q. Compare Watson's & Felix Model with SEL Model on a software development expected to involve 8 person year of Effort.

Ans → (a) Calculate no. of lines of code that can be produced :-

SEL :-

$$E = 1.4 L^{0.93}$$

$$\Rightarrow L = \left(\frac{E}{1.4}\right)^{1/0.93}$$

Now, we have given E = 8 person year

$$\Rightarrow E = 96 \text{ person Month}$$

$$L = \left(\frac{96}{1.4}\right)^{1/0.93} \text{ LOC}$$

$$= \left(\frac{960}{14}\right)^{1/0.93}$$

$$= \left(68.57\right)^{1/0.93}$$

$$= 94.26$$

Walston & Felix :-

$$E = 5.2\, L^{0.91}$$

$$L = \left(\frac{E}{5.2}\right)^{1/0.91}$$

$$L = \left(\frac{96}{5.2}\right)^{1/0.91}$$

$$L = 24.63 \text{ LOC.}$$

(b) Calculate the duration of development:

SEL :-

$$D = 4.6\, L^{0.26}$$

$$= 4.6\,(94.26)^{0.26}$$

$$= 14.9.997$$

$$= 150 \text{ Months.}$$

Walston & Felix :-

$$D = 4.1\, L^{0.36}$$

$$= 4.1\,(24.63)^{0.36}$$

$$= 4.1 \times 3.139.$$

$$= 12.558$$

$$= 12.6 \text{ Month}$$

(c) Calculate the productivity in (LOC/PM).

SEL :- $\frac{LOC}{Person\ Yes}$

$= \frac{94.26}{8} = 11.78$

Watson & Felix :-

$\frac{LOC}{Person\ Year} = \frac{24.63}{8}$

$= 3.07875$

COCOMO Model

Boehm proposed COCOMO (Constructive Cost Estimation Model) in 1981. It is one of the most generally used software estimation model of the world. COCOMO predicts the effort and schedule of a software product based on size of software.

Equation :-

$$E_i = a * (KDLOC) * b.$$

The value of a & b are constants.

KDLOC → Kilo Development line of Code.

Types of Project in COCOMO :-

1) Organic → A project can be called organic if the project deals with developing a well-understood application program, size of development team is reasonably small and all team members are experienced in developing similar method of project. In this requirement is fixed.

Example :- Simple Business System, Simple inventory management system, etc.

2) Semdetached :— The project which uses old concept and only ~~added~~ adding new features according to needs of the user is called semidetached project. In this the team members consist of ~~ex~~ mixture of experienced and unexperienced staff.

Example :- DBMS, OS, etc.

3) Embedded :- A development project is considered to be of embedded type if the software being developed is strongly coupled to complex hardware or if the ~~strict~~ stringent regulation on the operational methods exists.

Example :- ATM, Air Traffic Control.

Equation for Organic :-

$$Effort = 2.4 (KDLOC)\ 1.05\ PM.$$

$$Tdev = 2.5 (Effort)\ 0.38\ Months$$

Equation for Semidetached :-

$$Effort = 3.0 (KDLOC)\ 1.12\ PM$$

$$Tdev = 2.5 (Effort)\ 0.35\ Months.$$

Equation for Embeded :-

$$Effort = 3.6 (KDLOC)\ 1.20\ PM$$

$$Tdev = 2.5 (Effort)\ 0.32\ Months.$$

* Effort → total effort required to develop the software product expressed in PM.

* Tdev → Estimated time to develop software.

According to Boehm, Software cost estimation should be done through three stages :-

1. Basic Model
2. Intermediate Model
3. Detailed Model.

I) Basic COCOMO Model :- It provides the accurate size of ~~product~~ project parameters.

Expression :-

$$Effort = a_1 * (KLOC)\ a_2\ PM$$

$$Tdev = b_1 * (effort)\ b_2\ Months.$$

a_1, a_2, a_3, a_4 → Constants.

II) Intermediate Model :- The basic COCOMO model consider that the effort is only a function of number of lines of code and some constants calculated according to various software systems.

The Intermediate COCOMO Model refines the initial cost by using a set of 15 cost drivers based on various attributes of software engineering.

Expression :-

$$Effort = a_i (KLOC)^{b_i} EAF.$$
$$D = c_i (Effort)^{d_i}$$

EAF → Effort adjustment factor.

III) Detailed COCOMO Model :- It incorporates all qualities of the standard version with an assessment of the cost driver's effect on each method of the software engineering process.

Six phases of detailed COCOMO are :-

(a) Planning & requirements
(b) System structure
(c) Complete structure
(d) Module code & Test
(e) Integration & test
(f) Cost Constructive Model.

Q Suppose a project was estimated to be 400 KLOC. Calculate the effort and development time for each of the three model i.e. organic, semidetached & embedded.

Ans → Basic COCOMO eqn are :-

$$E = a_1 (KLOC)^{a_2} \text{ PM}$$

$$T_{dev} = b_1 (E)^{b_2} \text{ Months.}$$

Estimated size of project = 400 KLOC.

(i) Organic Mode :-

$$E = 2.4 * (400)^{1.05} = 1295.31 \text{ PM}$$

$$D = 2.5 * (1295.31)^{0.38} = 38.07 \text{ M}$$

(ii) Semi detached Mode :-

$$E = 3.0 * (400)^{1.12} = 2462.79 \text{ PM}$$

$$D = 2.5 * (2462.79)^{0.35} = 38.45 \text{ Months}$$

(iii) Embedded Mode :-

$$E = 3.6 * (400)^{1.20} = 4772.81 \text{ PM}$$

$$D = 2.5 * (4772.81)^{0.32} = 38 \text{ PM.}$$

Risk Management

* Risk :- Risk is a problem that could cause ~~one~~ some loss or threaten the progress of the project, but which has not happened yet.

Risk Management is the system of identifying addressing and eliminating these problems before they can damage the project.

Classification of Risks :-
1. Project Risks
2. Technical Risks
3. Business Risks.

1. Project Risks :- It concern with different form of budgetary, schedule, personnel, resource and customer related problems.

2. Technical risks :- It concerns with potential method implementation, interfacing, testing and maintainence issue. Most technical risk appear due to the development team's insufficient knowledge about the project.

3. Business Risk :- It contain risk of building an excellent product that no one need, losing budgetary & personnel commitments.

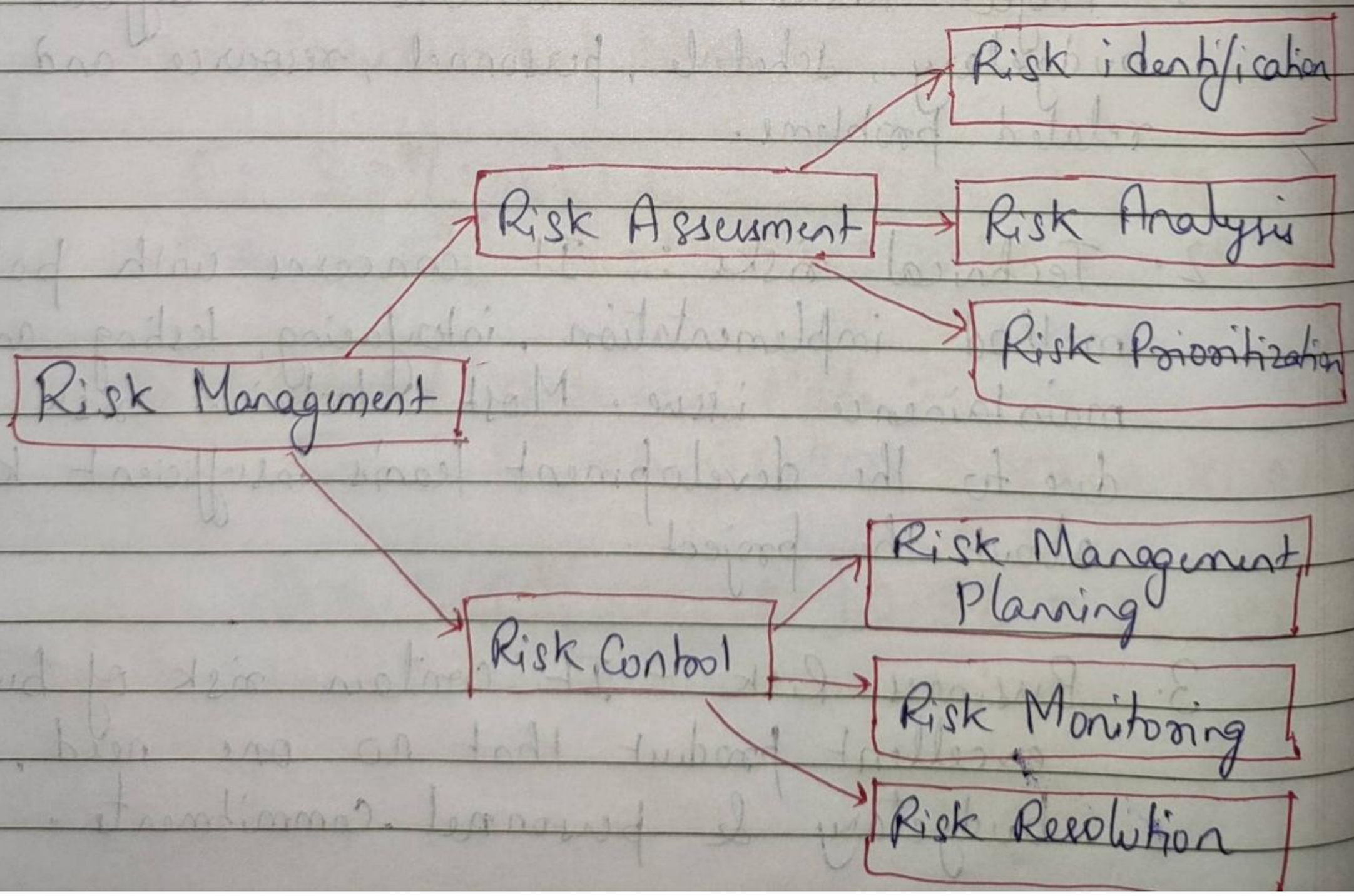
Other Risk Category :-

1) known risks
2) Predictable risk
3) Unpredictable risk

Principle of Risk Management :-

1) Global Perspective
2) Take a forward-looking view.
3) Open Communication
4) Integrated management
5) Continous process.

Risk Management Activities :-



Software Configuration Management :-

Software configuration is a combination of all the processes which make an output. As software development progresses, the number of Software configuration elements grow rapidly.

These need to managed for their proper working and that's where we require Software configuration management.

Therefore SCM is the discipline which :-
identify change.
Monitor and control change
Ensure the proper implementation of change
Auditing and reporting.

Configuration Management is a technique of identifying, organizing and controlling modification to software being built by a programming team.

Importance :-

1) It provide tools to ensure that changes are being properly implemented.
2) It has capability to describe & store various consituent of software.
3) It helps in controlling and managing the access to various SCIs. e.g. preventing team members looking for same system at same time.

SCM PROCESS:-

It involves the following activities:

1. **Identification and Establishment –** Identifying the configuration items from products that compose baselines at given points in time (a baseline is a set of mutually consistent Configuration Items, which has been formally reviewed and agreed upon, and serves as the basis of further development). Establishing relationship among items, creating a mechanism to manage multiple level of control and procedure for change management system.
2. **Version control –** Creating versions/specifications of the existing product to build new products from the help of SCM system. A description of version is given below:
3. **Change control –** Controlling changes to Configuration items (CI).
4. **Configuration auditing –** A software configuration audit complements the formal technical review of the process and product. It focuses on the technical correctness of the configuration object that has been modified. The audit confirms the completeness, correctness and consistency of items in the SCM system and track action items from the audit to closure.
5. **Reporting –** Providing accurate status and current configuration data to developers, tester, end users, customers and stakeholders through admin guides, user guides, FAQs, Release notes, Memos, Installation Guide, Configuration guide etc .

Software Project Planning:-

To plan a successful software project, we must understand:

o Scope of work to be completed

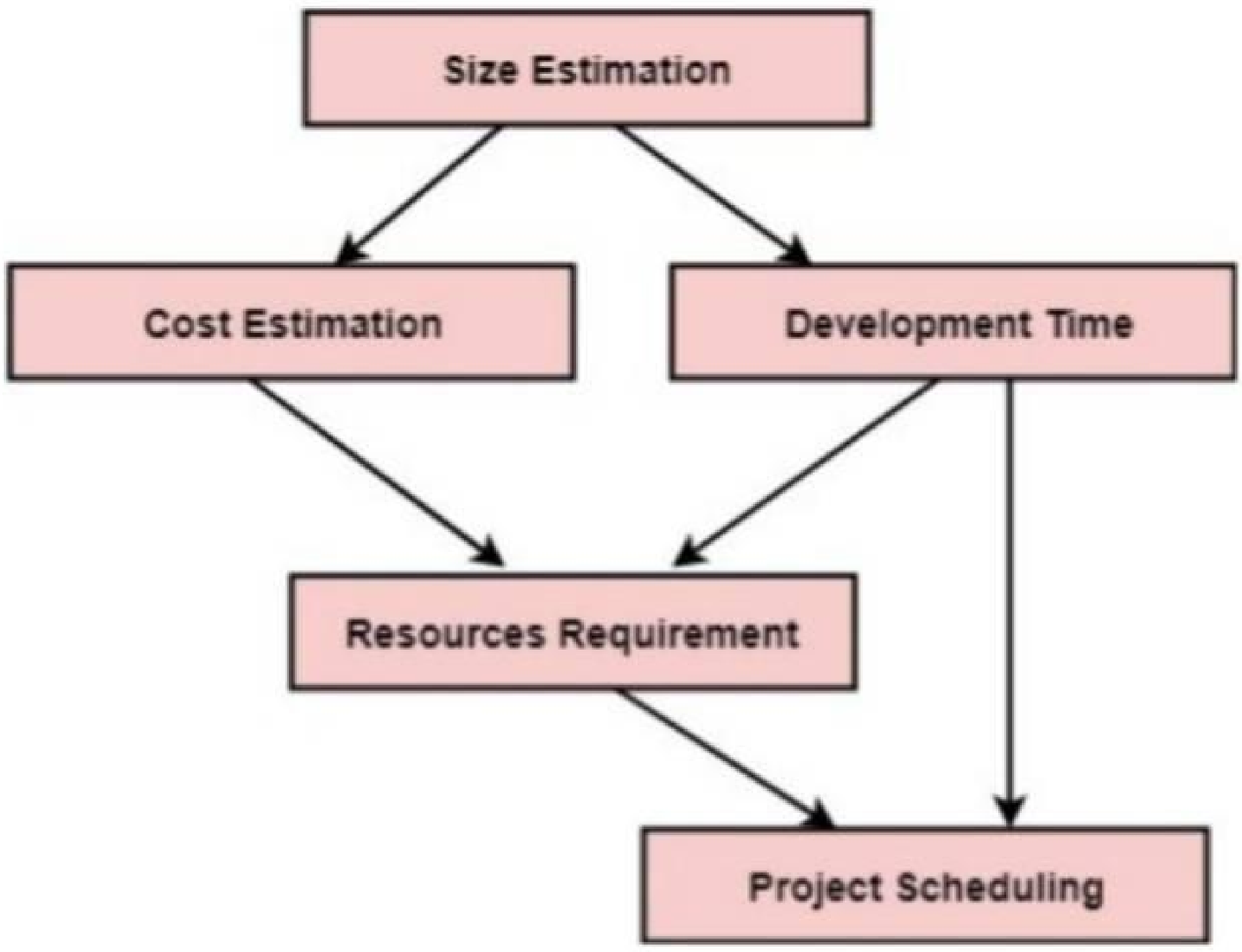
o Risk analysis

o The resources mandatory

o The project to be accomplished

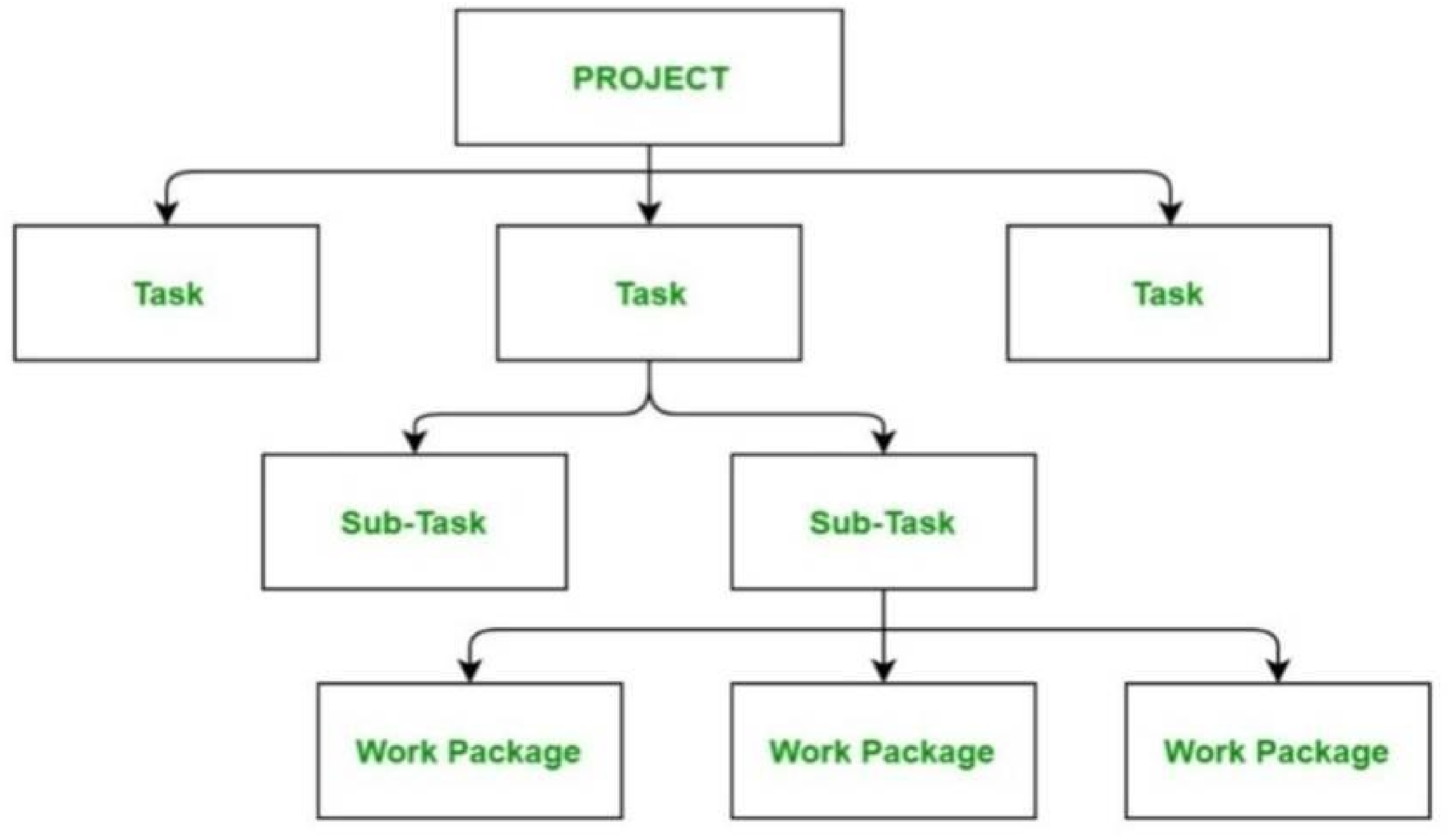
o Record of being followed

Software Project planning starts before technical work start. The various steps of planning activities are:



Work Breakdown Structure (WBS)

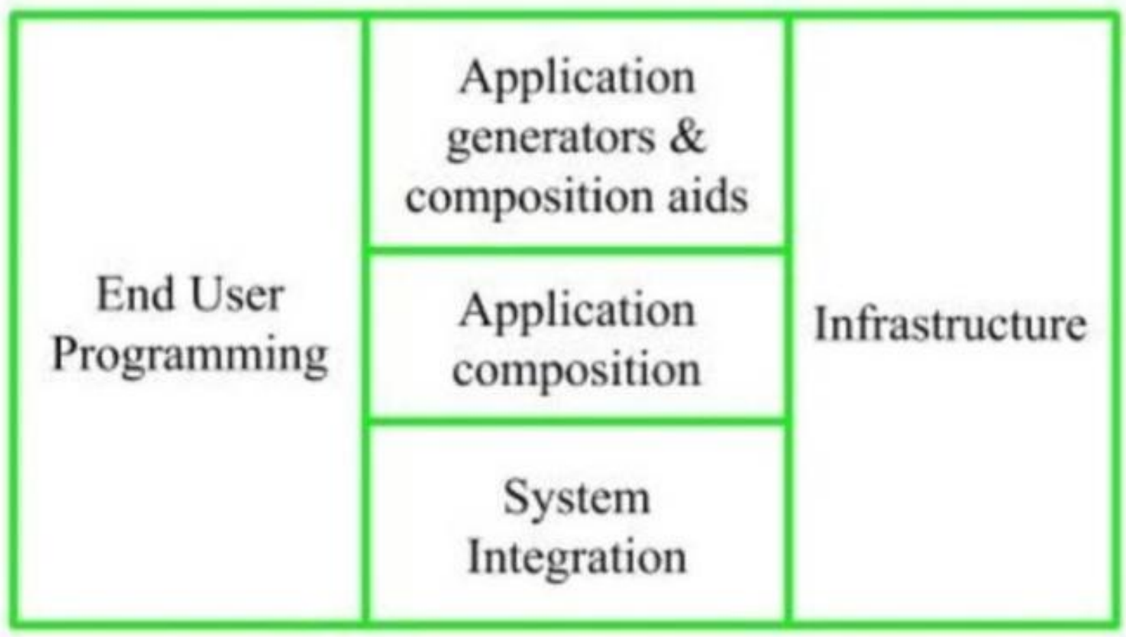
A Work Breakdown Structure includes dividing a large and complex project into simpler, manageable and independent tasks. The root of this tree (structure) is labeled by the Project name itself. For constructing a work breakdown structure, each node is recursively decomposed into smaller sub-activities, until at the leaf level, the activities becomes undividable and independent. It follows a Top-Down approach.



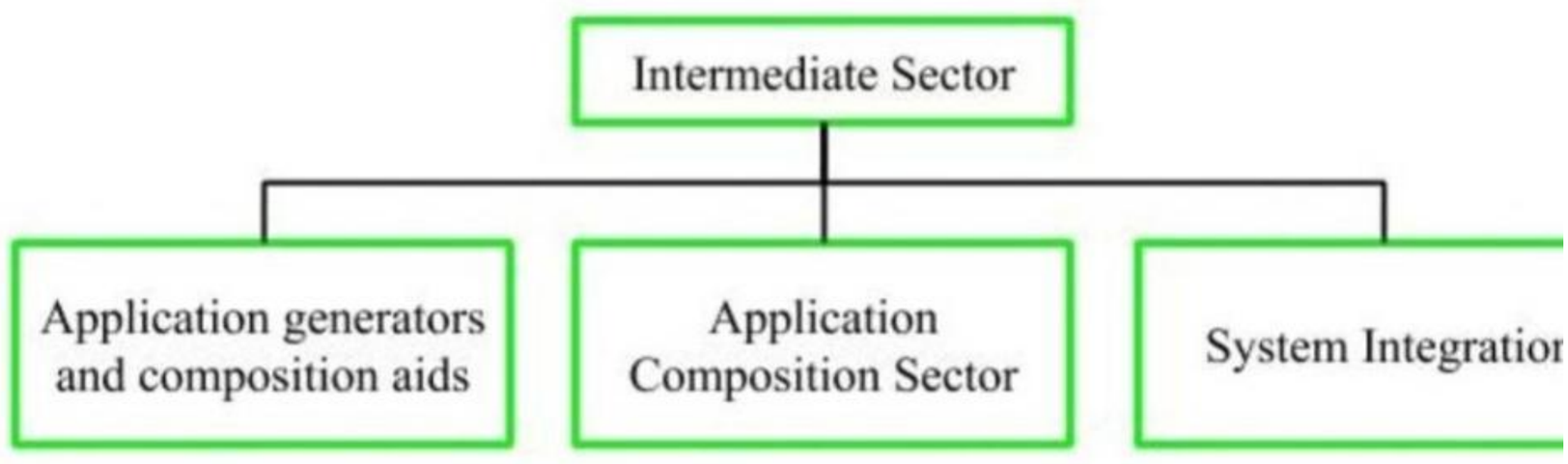
COCOMO-II

COCOMO-II is the revised version of the original COCOMO (Constructive Cost Model) and is developed at University of Southern California. It is the model that allows one to estimate the cost, effort and schedule when planning a new software development activity.

It consists of three sub-models:



1. End User Programming:
Application generators are used in this sub-model. End user write the code by using these application generators.
Example – Spreadsheets, report generator, etc.
2. Intermediate Sector:

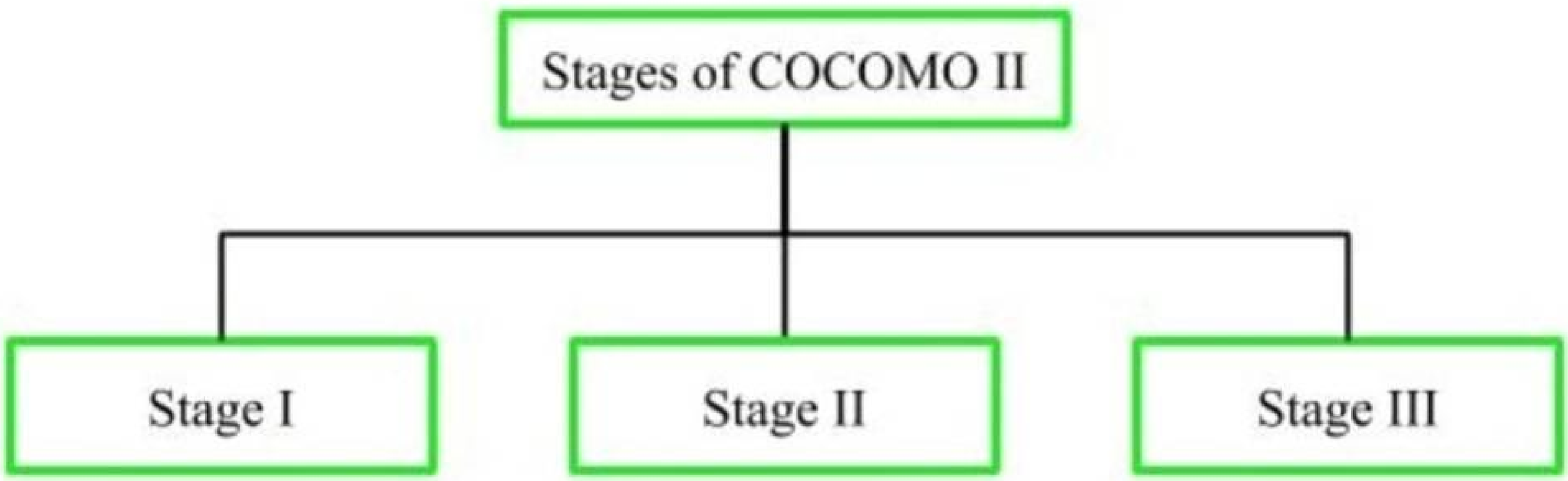


- **(a). Application Generators and Composition Aids –**
  This category will create largely prepackaged capabilities for user programming Their product will have many reusable components. Typical firms operating in th sector are Microsoft, Lotus,
  Oracle, IBM, Borland, Novell.
- **(b). Application Composition Sector –**
  This category is too diversified and to be handled by prepackaged solutions. It includes GUI, Databases, and domain specific components such as financial, medical or industrial process control packages.
- **(c). System Integration –**
  This category deals with large scale and highly embedded systems.

3. Infrastructure Sector:

This category provides infrastructure for the software development like Operating System, Database Management System, User Interface Management System, Networking System, etc.

Stages of COCOMO II:



1. **Stage-I:**
   It supports estimation of prototyping. For this it uses *Application Composition Estimation Model*. This model is used for the prototyping stage of application generator and system integration.
2. **Stage-II:**
   It supports estimation in the early design stage of the project, when we less kno about it. For this it uses *Early Design Estimation Model*. This model is used in ea design stage of application generators, infrastructure, and system integration.
3. **Stage-III:**
   It supports estimation in the post architecture stage of a project. For this it uses *Post Architecture Estimation Model*. This model is used after the completio of the detailed architecture of application generator, infrastructure, and system integration.

Difference between COCOMO 1 & COCOMO 2

COCOMO 1

- It is used in waterfall model of software development cycle.
- It helps give estimates required for the efforts and schedule.
- It is based on the linear reuse formula.
- It is based on the assumption of reasonable stable requirements.

- The effort equation has an exponent that is figured out using 3 developmen modes.
- The development begins after the requirements are assigned to software.
- The numbers of sub-models are 3.
- It has 15 cost drivers assigned to it.
- The size of the software is measured in terms of lines of code.

COCOMO 2

- It is used in non-sequential, rapid development of models.
- It also helps reuse the models of software.
- It helps provide estimates which represent a standard deviation near the mos likely estimate.
- It is based on the non-linear formula of reuse.
- It is based on the reuse model that focuses on effort required to understand an estimate.
- The effort equation has an exponent that is determined by 5 scale factors.
- It uses spiral development method.
- The numbers of sub-models required are 4.
- It is assigned with 17 cost drivers.
- The size of software is stated in terms of object points, function points and line of code.