Module – 2: Project Life cycle and Estimation: Review

Software Process

- A software process is the set of activities and associated outcome that produce a software product.
- Software engineers mostly carry out these activities. These are four key process activities, which are common to all software processes. ***These activities are:***

1. **Software specifications:** The functionality of the software and constraints on its operation must be defined.
2. **Software development:** The software to meet the requirement must be produced.
3. **Software validation:** The software must be validated to ensure that it does what the customer wants.
4. **Software evolution:** The software must evolve to meet changing client needs.

Factors in choosing a software process

- Choosing the right software process model for your project can be difficult.
- If you know your requirements well, it will be easier to select a model that best matches your needs.
- The following factors in mind when selecting your software process model:

1. **Project requirements :** Before you choose a model, take some time to go through the project requirements and clarify them alongside your organization's or team's expectations.
2. **Project size :** Consider the size of the project you will be working on.
   Larger projects mean bigger teams, so you'll need more extensive and elaborate project management plans.
1. **Project complexity:** Complex projects may not have clear requirements.
   The requirements may change often, and the cost of delay is high.
   Ask yourself if the project requires constant monitoring or feedback from the client.
1. **Cost of delay:** Is the project highly time-bound with a huge cost of delay, or are the timelines flexible?
1. **Customer involvement:** Do you need to consult the customers during the process?
   Does the user need to participate in all phases?
1. **Familiarity with technology:** This involves the developers' knowledge and experience with the project domain, software tools, language, and methods needed for development.
2. **Project resources:** This involves the amount and availability of funds, staff, and other resources.

Software Process Model

- A software process model is a specified definition of a software process, which is presented from a particular perspective.
- Models, by their nature, are a simplification, so a software process model is an abstraction of the actual process, which is being described.
- Process models may contain activities, which are part of the software process, software product, and the roles of people involved in software engineering.

Some examples of the types of software process models that may be produced are:

1. **A workflow model:**

- This shows the series of activities in the process along with their inputs, outputs and dependencies.
- The activities in this model perform human actions.

2. **A dataflow or activity model:**

- It represents the process as a set of activities, carries out some data transformations.
- It shows how the input to the process, such as a specification is converted to an output such as a design.
- The activities here may be at a lower level than activities in a workflow model.
- They may perform transformations carried out by people or by computers.

3. **A role/action model:**

- This means the roles of the people involved in the software process and the activities for which they are responsible.

There are several various general models or paradigms of software development:

- **The waterfall approach:**

- This takes the above activities and produces them as separate process phases such as requirements specification, software design, implementation, testing, and so on.
- After each stage is defined, it is "signed off" and development goes onto the different stage.

- **Evolutionary development:**

- This method interleaves the activities of specification, development, and validation.
- An initial system is rapidly developed from a very abstract specification.

- **Formal transformation:**

- This method is based on producing a formal mathematical system specification and transforming this specification, using mathematical methods to a program.
- These transformations are 'correctness preserving.' This means that you can be sure that the developed programs meet its specification.

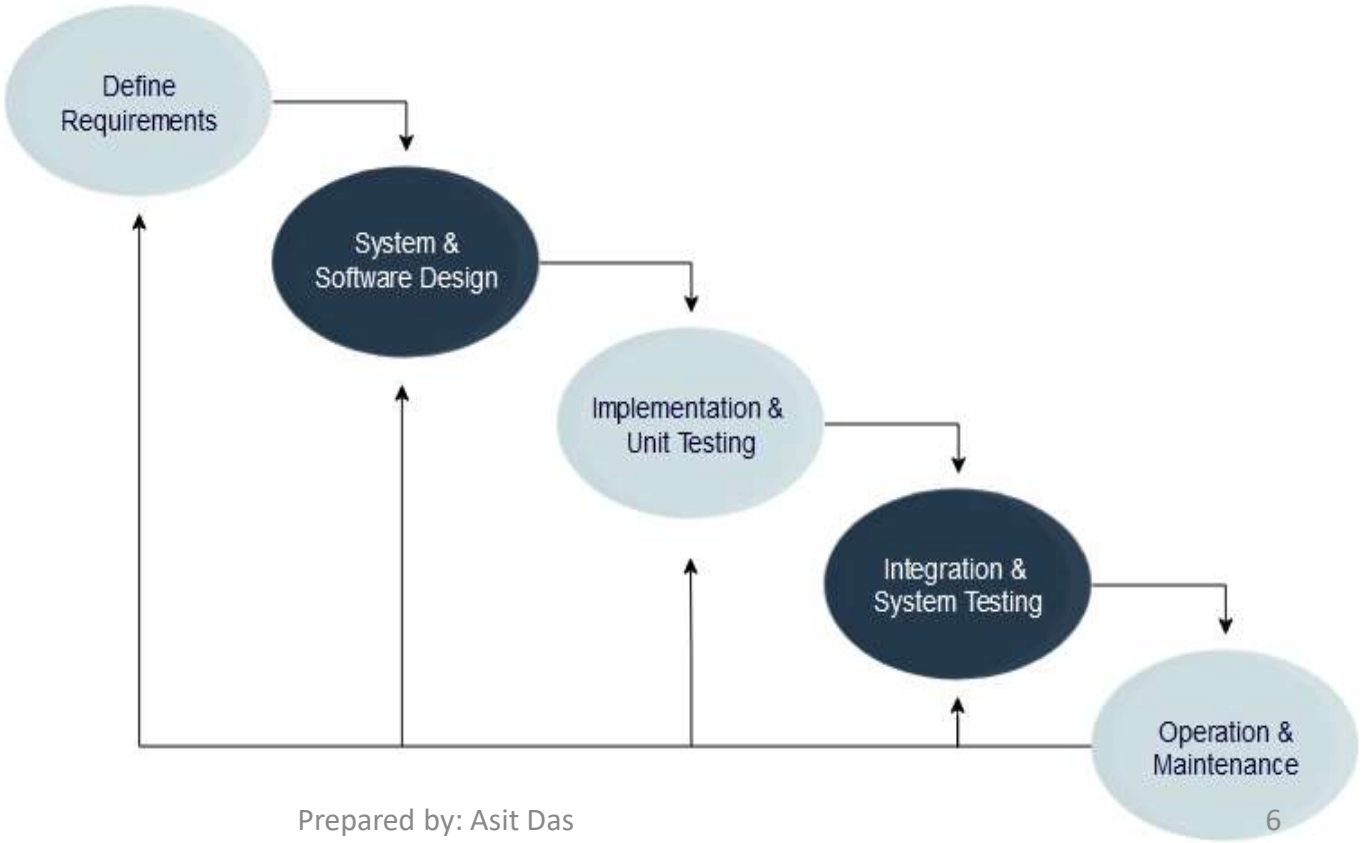
- **System assembly from reusable components:**

- This method assumes the parts of the system already exist.
- The system development process target on integrating these parts rather than developing them from scratch.

Types of software process models

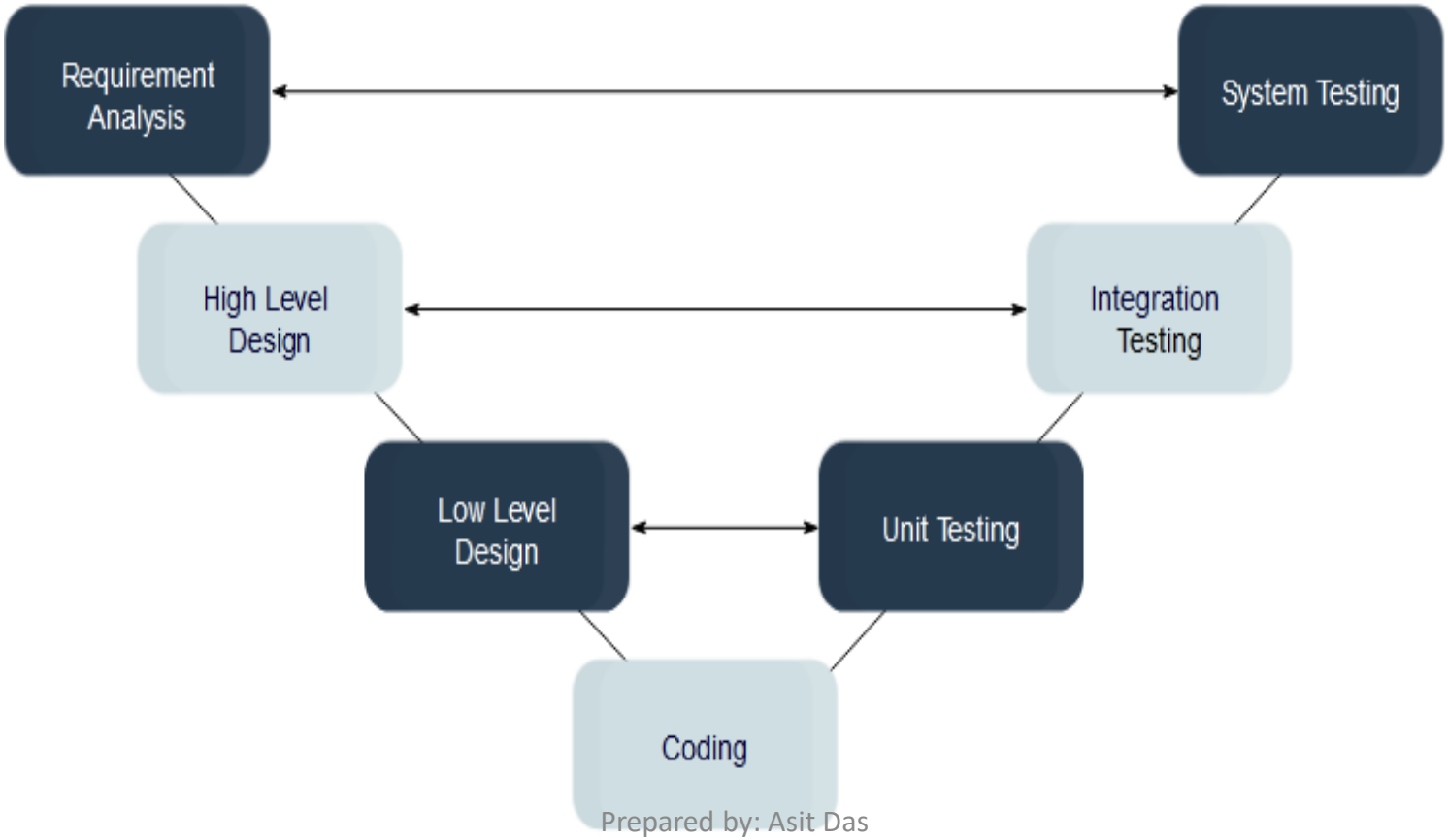
1. Waterfall Model

- The waterfall model is a **sequential, plan driven-process** where you must plan and schedule all your activities before starting the project. Each activity in the waterfall model is represented as a separate phase arranged in linear order.
- It has the following phases:
- Requirements
- Design
- Implementation
- Testing
- Deployment
- Maintenance



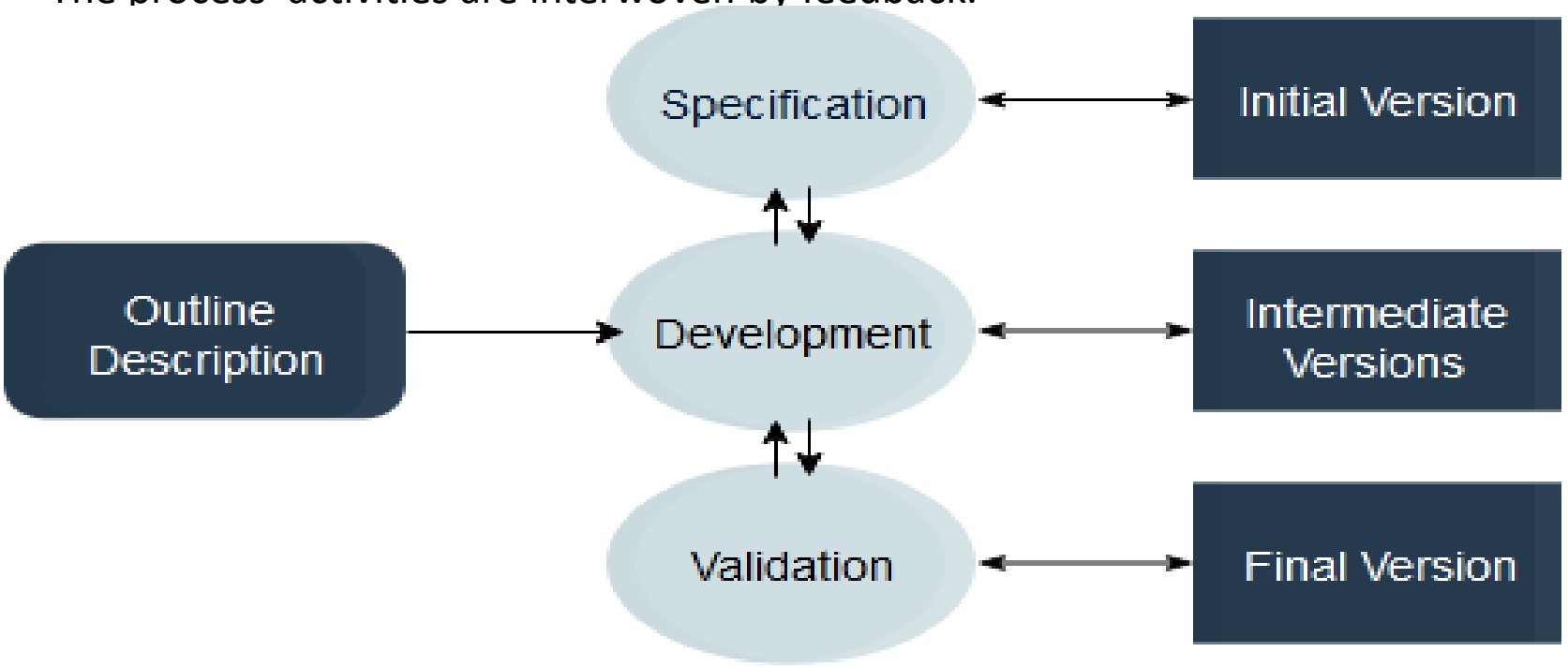
2. V Model

- The V model (Verification and Validation model) is an extension of the waterfall model.
- All the requirements are gathered at the start and cannot be changed.
- You have a corresponding testing activity for each stage.
- For every phase in the development cycle, there is an **associated testing phase.**



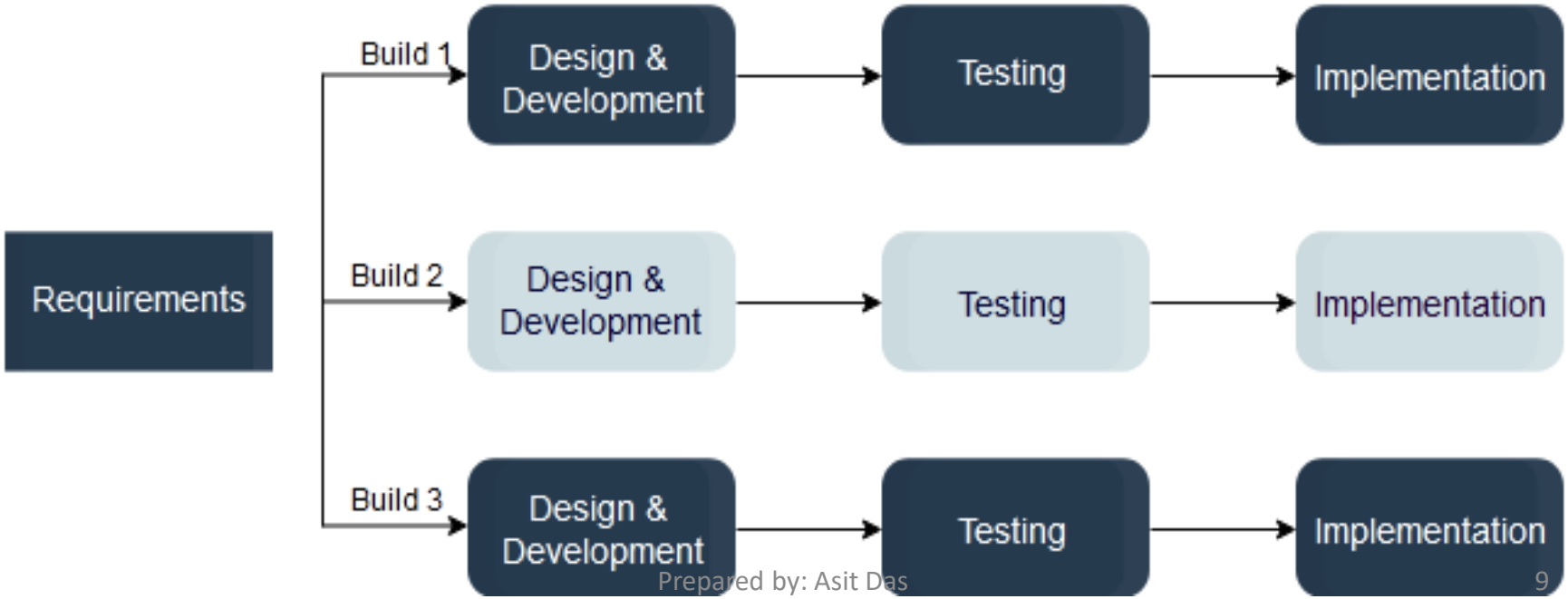
3. Incremental Model

- The incremental model divides the system's functionality into **small increments** that are delivered one after the other in quick succession.
- The most important functionality is implemented in the initial increments.
- The subsequent increments expand on the previous ones until everything has been updated and implemented.
- Incremental development is based on developing an initial implementation, exposing it to user feedback, and evolving it through new versions.
- The process' activities are interwoven by feedback.



4. Iterative Model

- The iterative development model develops a system through **building small portions** of all the features.
- This helps to meet initial scope quickly and release it for feedback.
- In the iterative model, you start off by implementing a small set of the software requirements.
- These are then **enhanced iteratively** in the evolving versions until the system is completed.
- This process model starts with part of the software, which is then implemented and reviewed to identify further requirements.

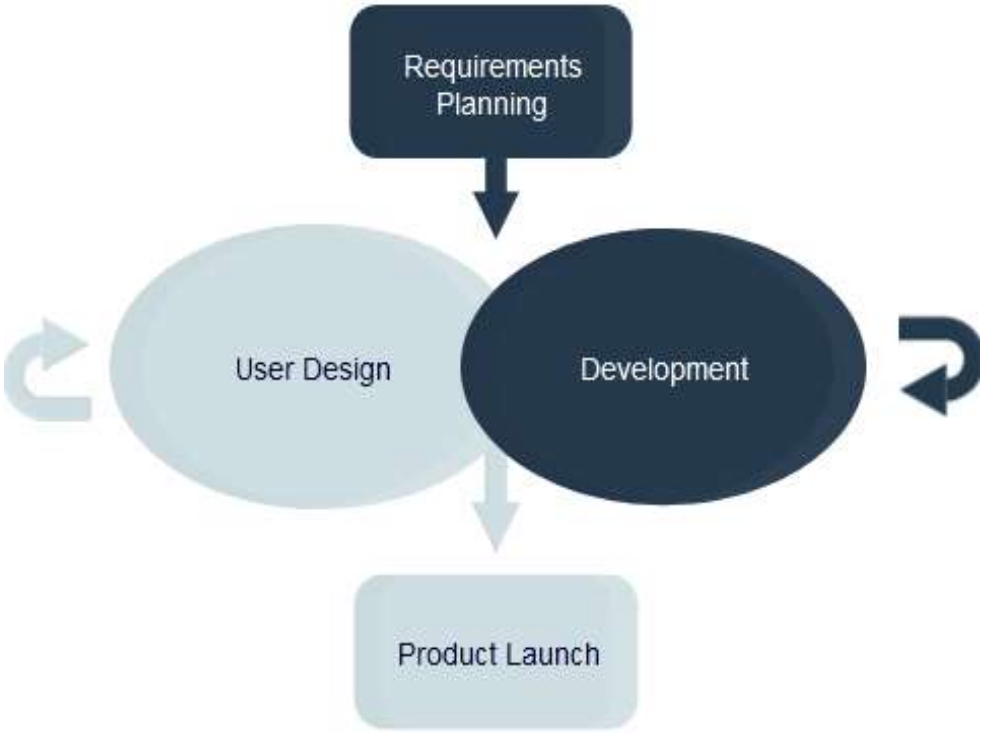


5. RAD Model

- The Rapid Application Development (RAD model) is based on iterative development and prototyping with **little planning involved**.

It involves the following phases:

- *Business modeling*
- *Data modeling*
- *Process modeling*
- *Application generation*
- *Testing and Turnover*



The RAD model accommodates changing requirements, reduces development time, and increases the reusability of components.

But it can be complex to manage.

Therefore, the RAD model is great for systems that need to be produced in a short time and have known requirements.

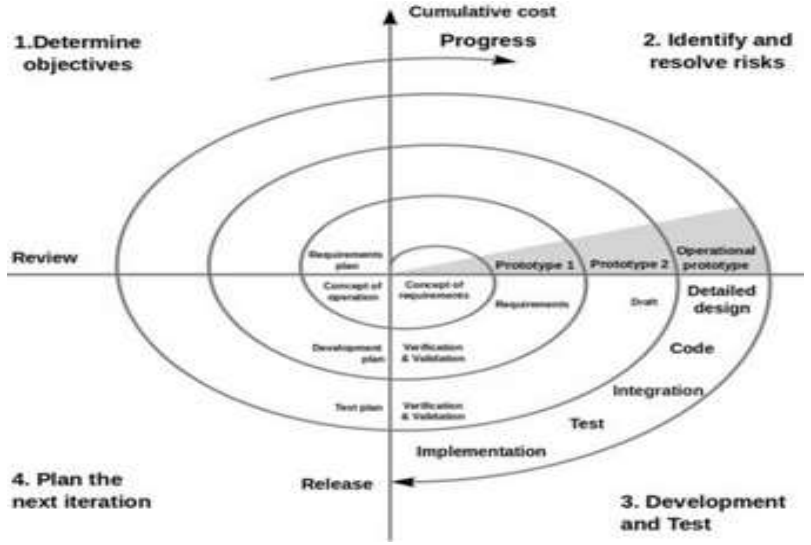
The RAD concept focuses on gathering requirements using focus groups and workshops, reusing software components, and informal communication.

6. Spiral Model

- The spiral model is a risk driven iterative software process model.
- This model delivers projects in loops.
- Unlike other process models, its steps aren't activities but **phases** for addressing whatever problem has the greatest risk of causing a failure.

Following phases for each cycle :

- Address the highest-risk problem and determine the objective and alternate solutions
- Evaluate the alternatives and identify the risks involved and possible solutions
- Develop a solution and verify if it's acceptable
- Plan for the next cycle



7. Agile model

- The agile process model encourages continuous iterations of development and testing.
- Each incremental part is developed over an iteration, and each iteration is designed to be small and manageable .
- Thus it can be completed within a few weeks.

- Each iteration focuses on implementing a small set of features completely.
- It involves customers in the development process and minimizes documentation by using informal communication.

Agile development considers the following:

- ***Requirements are assumed to change***
- ***The system evolves over a series of short iterations***
- ***Customers are involved during each iteration***
- ***Documentation is done only when needed***

--- Though agile provides a very realistic approach to software development, it isn't great for complex projects.

---- It can also present challenges during transfers as there is very little documentation.

---- Agile is great for projects with **changing requirements**.

Some commonly used agile methodologies include:

- **Scrum:**
  - One of the most popular agile models, Scrum consists of iterations called sprints.
  - Each sprint is between 2 to 4 weeks long and is preceded by planning.
  - Any changes cannot possible after the sprint activities have been defined.
- **Extreme Programming (XP):**
  - With Extreme Programming, an iteration can last between 1 to 2 weeks.
  - XP uses pair programming, continuous integration, test-driven development and test automation, small releases, and simple software design.
- **Kanban:**
  - Kanban focuses on visualizations, and if any iterations are used they are kept very short.
  - Use the Kanban Board that has a clear representation of all project activities and their numbers, responsible people, and progress.

8. Extreme Programming (XP)

- Extreme programming (XP) is one of the most important software development frameworks of Agile models.
- It is used to improve software quality and responsiveness to customer requirements.
- The extreme programming model recommends taking the best practices that have worked well in the past in program development projects to extreme levels.

Good practices need to be practiced in extreme programming:

- Some of the good practices that have been recognized in the extreme programming model and suggested to maximize their use are given below:

- **Code Review:** Code review detects and corrects errors efficiently. It suggests pair programming as coding and reviewing of written code carried out by a pair of programmers who switch their works between them every hour.
- **Testing:** Testing code helps to remove errors and improves its reliability. XP suggests test-driven development (TDD) to continually write and execute test cases. In the TDD approach test cases are written even before any code is written.
- **Incremental development:** Incremental development is very good because customer feedback is gained and based on this development team comes up with new increments every few days after each iteration.
- **Simplicity:** Simplicity makes it easier to develop good quality code as well as to test and debug it.
- **Design:** Good quality design is important to develop good quality software. So, everybody should design daily.
- **Integration testing:** It helps to identify bugs at the interfaces of different functionalities. Extreme programming suggests that the developers should achieve continuous integration by building and performing integration testing several times a day.

Basic principles of Extreme programming:

Some of the basic activities that are followed during software development by using the XP model are given below:

- **Coding:**

- The concept of coding which is used in the XP model is slightly different from traditional coding.
- Coding activity includes drawing diagrams (modeling) that will be transformed into code, scripting a web-based system, and choosing among several alternative solutions.

- **Testing:**

- XP model gives high importance to testing .
- Quality is the primary factor to develop fault-free software.

- **Listening:**

- The developers need to carefully listen to the customers if they have to develop good quality software.
- Sometimes programmers may not have the depth knowledge of the system to be developed.
- So, the programmers should understand properly the functionality of the system and they have to listen to the customers.

- **Designing:**

- Without a proper design, a system implementation becomes too complex and very difficult to understand the solution thus making maintenance expensive.
- A good design results elimination of complex dependencies within a system.
- So, effective use of suitable design is emphasized.

- **Feedback:**

- One of the most important aspects of the XP model is to gain feedback to understand the exact customer needs.
- Frequent contact with the customer makes the development effective.

- **Simplicity:**

- The main principle of the XP model is to develop a simple system that will work efficiently in the present time.
- Rather than trying to build something that would take time and may never be used.
- It focuses on some specific features that are immediately needed, rather than engaging time and effort on speculations of future requirements.

Applications of Extreme Programming (XP):

Some of the projects that are suitable to develop using the XP model are given below:

- **Small projects:**
  - XP model is very useful in small projects consisting of small teams as the face-to-face meeting is easier to achieve.
- **Projects involving new technology or Research projects:**
  - This type of project face changing requirements rapidly and technical problems.
  - So XP model is used to complete this type of project.

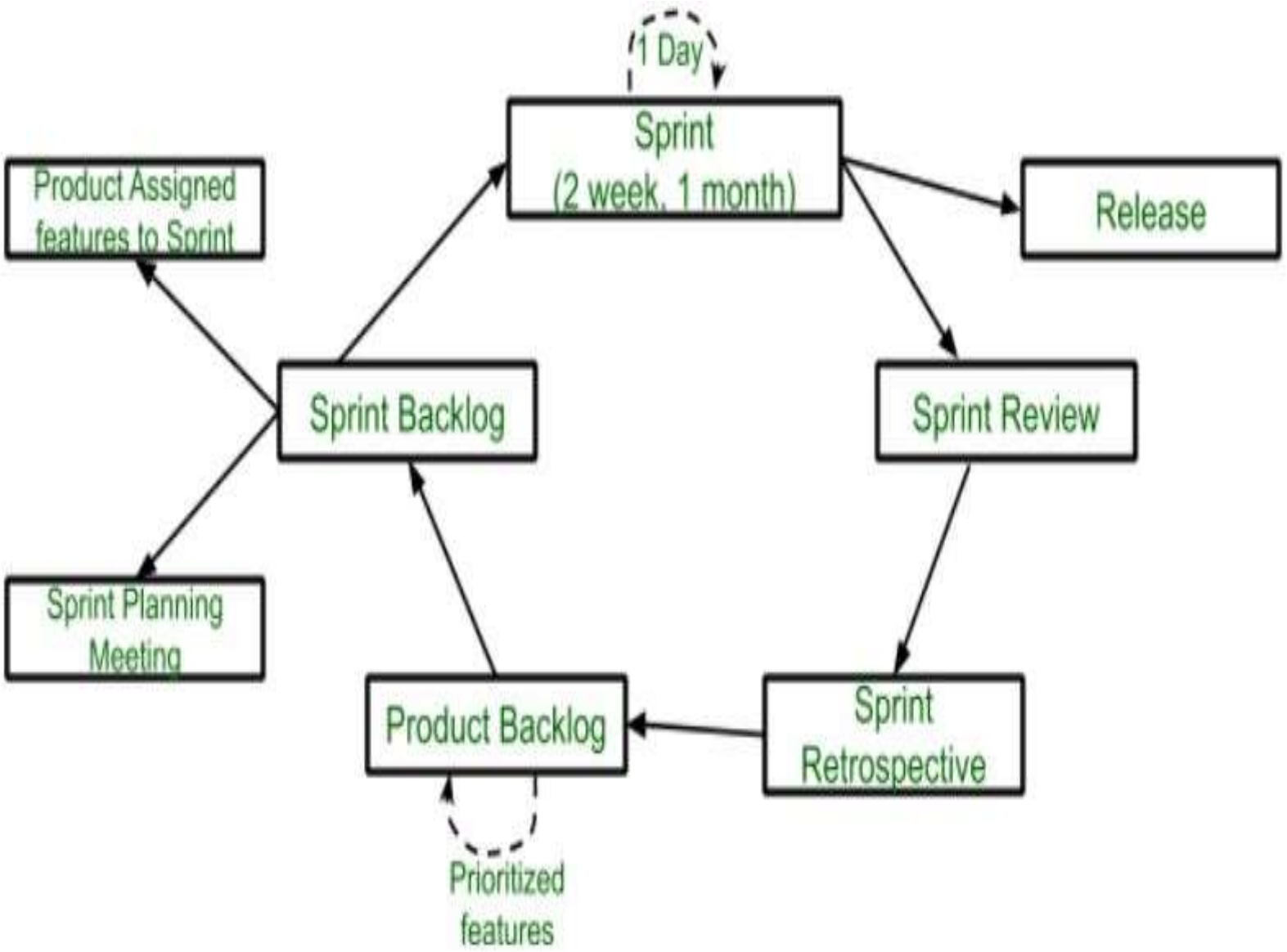
SCRUM

- Scrum is an Agile project management methodology involving a small team led by a Scrum master,
- Main job is to remove all obstacles to getting work done.
- Work is done in short cycles called sprints, and the team meets daily to discuss current tasks and any roadblocks that need clearing.
- Scrum is a method for managing projects that allows for rapid development and testing, especially within a small team.
- Scrum uses **Iterative process**.

Silent features of Scrum are:

- **Scrum is light-weighted framework**
- **Scrum emphasizes self-organization**
- **Scrum is simple to understand**
- **Scrum framework help the team to work together**

Lifecycle of Scrum:



- **Sprint:**
  - A Sprint is a time-box of one month or less.
  - A new Sprint starts immediately after the completion of the previous Sprint.
- **Release:**
  When the product is completed then it goes to the Release stage.
- **Sprint Review:**
  If the product still have some non-achievable features then it will be checked in this stage and then the product is passed to the Sprint Retrospective stage.
- **Sprint Retrospective:**
  In this stage quality or status of the product is checked.
- **Product Backlog:**
  According to the prioritize features the product is organized.
- **Sprint Backlog:**
  Sprint Backlog is divided into two parts
  - Product assigned features to sprint and
  - Sprint planning meeting.

Advantage of using Scrum framework:

- Scrum framework is fast moving and money efficient.
- Scrum framework works by dividing the large product into small sub-products. It's like a divide and conquer strategy
- In Scrum customer satisfaction is very important.
- Scrum is adaptive in nature because it have short sprint.
- As Scrum framework rely on constant feedback therefore the quality of product increases in less amount of time

Disadvantage of using Scrum framework:

- Scrum framework do not allow changes into their sprint.
- Scrum framework is not fully described model.
- If you adopt it you need to fill in the framework with your own details like Extreme Programming(XP), Kanban, DSDM.
- It can be difficult for the Scrum to plan, structure and organize a project that lacks a clear definition.
- The daily Scrum meetings and frequent reviews require substantial resources.

Managing Interactive Processes

Following steps are essential for managing interactive processes in software management:

1. **Start with a project charter**
2. **Write a brief scope statement**
3. **Develop a thorough project management plan**
4. **Direct and manage the project work**
5. **Monitor and control the project work**
6. **Close the project by trying up all the loose ends.**

Managing People

- **Act as project leader**
- **Liaison with stakeholders**
- **Managing human resources**
- **Setting up reporting hierarchy etc.**

Managing Project

- Defining and setting up project scope
- Managing project management activities
- Monitoring progress and performance
- Risk analysis at every phase
- Take necessary step to avoid or come out of problems
- Act as project spokesperson

Basics of Software Estimation:

- **Activities involved in Software Estimation:**

1. Projects planning (Estimation determines how much money, effort, resources, and time it will take to build a specific system or product)

2. Scope and feasibility (The functions and features that are to be delivered to end users. The data that are input to and output from the system.

The "content" that is presented to users as a consequence of using the software)

3. Project resources (Each resource is specified with: A description of the resource. A statement of availability, time when the resource will be required. The duration of time that the resource will be applied Time window)

- **Estimation of project cost and effort (The accuracy of a software project estimate is predicated on:** The degree to which the planner has properly estimated the size (e.g., KLOC) of the product to be built. The ability to translate the size estimate into human effort, calendar time, and money)

- **Decomposition techniques(Before an estimate can be made and decomposition techniques applied,** the planner must Understand the scope of the software to be built Generate an estimate of the software's size)

- **Empirical estimation models(Estimation models for computer software use empirically derived formulas to predict effort as a function of LOC (line of code) or FP(function point).**Resultant values computed for LOC or FP are entered into an estimation model)

Cost benefit Analysis

- Cost benefit analysis (CBA) is nothing but to develop a powerful or an efficient system.
- While thinking out on CBA, we need to find out factors that really affect benefits and costs of system.
- In developing cost estimates for a system, we need to consider some of cost elements.
- Some elements among them are hardware, personnel, facility, operating and supply cost.

The following are the cost factors :

1. **Hardware cost –**
   - Hardware cost includes actual purchase and peripherals (external devices) that are connected to computer. For example, printer, disk drive etc.
2. Actually, finding actual cost of hardware is generally more difficult , when system is shared by various users so as to compared to a system which dedicated stand alone .
3. In some case, best way is to treat it as operating cost.

2. Personnel costs –

- Personnel costs includes EDP staff salaries and benefits as well as pay for those who are involved in process of development of system.
- Cost occurred during development of system which are one time costs and are also called development cost.
- Once system is installed, cost of operating and maintaining system becomes recurring cost that one has to pay very frequently based on requirement.

3. Facility cost –

- Facility cost is amount of money that is spent in preparation of a site that is physical where application or computer will be in operation.
- This includes wiring, flooring, lighting and air conditioning.
- These costs are treated as one- time costs and are included into overall cost estimate of candidate system.

4. Operating costs –

- These includes all costs associated with day-to-day(everyday) operation of system and amount depends on number of shifts, nature of applications.
- There are various ways of covering operating costs. One approach is to treat operating costs as an overhead. Another approach is to charge money from each authorized user for amount of processing they require from system.
- Amount charged is based on computer time or time they spend on system, staff time ad volume of output produced .

5. Supply costs –

- Supply cost are variable costs that increase with increased use of paper, disks and like.
- They should be estimated and included in overall cost of system.

- *Benefits may be tangible and intangible, direct or indirect.*
- ***Two major benefits are improving performance and minimizing cost of processing of system.***

- The **performance category emphasizes improvement in accuracy** of or access to information and easier access to system by authorized users.
- Minimizing costs through an efficient system – **error control or reduction of staff-** is a benefit that should be measured and included in cost/benefit analysis.

The determination of costs and benefit entails following steps :

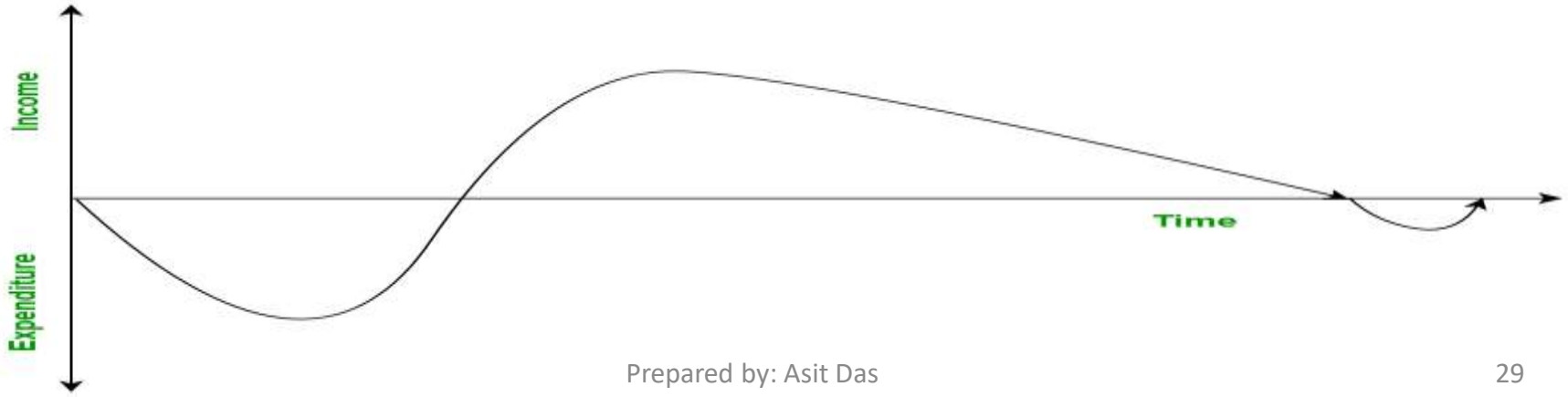
- **Identify the costs and benefits pertaining to given project.**
- **Categorize the various costs and benefits for analysis.**
- **Select a method of evaluation.**
- **Interpret the results of the analysis.**
- **Take action.**

Cash Flow Forecasting:

- **Cash flow** is the movement of the money in and out of an organisation.
- It involves the expenditure and income of an organisation.

Cash Flow Forecasting:

- In simple words, it is the estimation of the cash flow over a period of time.
- It is important to do cash flow forecasting in order to ensure that the project has sufficient funds to survive.
- It gives an estimation that when income and expenditure will take place during the software project's life cycle.
- It must be done time to time especially for start-ups and small enterprises.
- However, if the cash flow of the business is more stable then forecasting cash flow weekly or monthly is enough.



Cash flow is of two types:

1. Positive Cash Flow:

If an organisation expects to receive income more than it spends then it is said to have a positive cash flow and

The company will never go low on funds for the software project's completion.

2. Negative Cash Flow:

If an organisation expects to receive income less than it spends then it is said to have a negative cash flow and

The company will go low on funds for the software project's completion in future.

Importance of Cash Flow Forecasting:

- It allows the management to plan the expenditures based upon the income in future.
- It helps the organization to analyses its expenditures and incomes.
- Makes sure that the company can afford to pay the employees and suppliers.
- Helps in financial planning.

Cost benefit evaluation Techniques

- The primary objective of Cost benefit analysis is to find out whether it is economically worthwhile to invest in the proposed project. If the return on investment is high, the project is considered economically worthwhile.

The common methods for Cost-Benefit Evaluation are:-

- ***Payback Technique***
- ***Present Value Technique***
- ***Net Present Technique***
- ***Net Benefit Technique***
- ***Break Even Technique***
- ***Cash Flow Technique***

1. Payback Technique:

- Payback technique defines the time required to recover the money spent on a project.
- **Formula**: **Payback Time = Overall cost outlay / Annual cash return+ installation period.**

2. Present Value Technique

- The payback technique has certain drawbacks. The value of today's money& tomorrow's money is not the same. In the payback method, today's cost is compared with tomorrow's benefits & thus the time value of money is not considered.
- The present value technique compared the present value to future values by considering the time value of invested money.
- The present value analysis determines how much money it is worthwhile investing now, in order to receive a given return in some year's time.
- **Formula : P=F/(1+r/n)**

Where **P is the present value, F is the future value , r is the rate of interest , n is the number of years**

3. Net Present Value Technique:

- This method takes into consideration the time value of money & attempts to calculate the return on investment by introducing the factor of the time element.
- The net present values of all inflows& outflows of cash occurring during the entire life of the project are determined separately for each year by discounting these flows by the firm's cost of capital.

4. Net Benefit Technique:

- In this method, the net benefit is calculated by subtracting the total estimated cost from the total estimated benefit.

5. Break Even Technique:

- In the Break-Even Method, the costs of current & new systems are compared to find the time when both are equal. This point is called a break-even point.
- The breakeven point is the time at which the cost of the new system equals the cost of the current one.

6. Cash Flow analysis Technique:

- Cash flow is a procedure designed to keep track of accumulated saving and expenditure on a regular basis.
- In cash flow analysis method, projected expenditure& costs are identified & totaled.
- The difference between the incoming savings& outgoing expenses is the cash flow.

Function point calculation

- Function point is an element of software development which helps to approximate the cost of development early in the process.
- It may measures functionality from user's point of view.

Counting Function Point (FP):

- **Step-1:** F = 14 * scale (Scale varies from 0 to 5) according to character of Complexity Adjustment Factor (CAF). Below table shows scale:

  0 - No Influence

  1 - Incidental

  2 - Moderate

  3 - Average

  4 - Significant

  5 - Essential

- **Step-2:** Calculate Complexity Adjustment Factor (CAF).

  CAF = 0.65 + (0.01 * F)

- **Step-3:** Calculate Unadjusted Function Point (UFP).

  TABLE (Required)

Function Units	Low	Avg	High
EI	3	4	6
EO	4	5	7
EQ	3	4	6
ILF	7	10	15
EIF	5	7	10

Multiply each individual function point to corresponding values in TABLE.

Step-4: Calculate Function Point.

FP = UFP * CAF

Example:

Given the following values, compute function point when all complexity adjustment factor (CAF) and weighting factors are average.

- User Input = 50
- User Output = 40
- User Inquiries = 35
- User Files = 6
- External Interface = 4

Explanation:

Step-1: As complexity adjustment factor is average (given in question), hence, scale = 3.

F = 14 * 3 = 42

Step-2: CAF = 0.65 + (0.01 * 42) = 1.07

Step-3: As weighting factors are also average (given in question) hence we will multiply each individual function point to corresponding values in TABLE.

UFP = (50*4) + (40*5) + (35*4) + (6*10) + (4*7)

Step-4: Function Point = 628 * 1.07 = 671.96

This is the required answer.

Types of FP Attributes

1. FPs of an application is found out by counting the number and types of functions used in the applications. Various functions used in an application can be put under five types, as shown in Table:

Measurements Parameters	Examples
1.Number of External Inputs(EI)	Input screen and tables
2. Number of External Output (EO)	Output screens and reports
3. Number of external inquiries (EQ)	Prompts and interrupts.
4. Number of internal files (ILF)	Databases and directories
5. Number of external interfaces (EIF)	Shared databases and shared routines.

All these parameters are then individually assessed for complexity.

2. FP characterizes the complexity of the software system and hence can be used to depict the project time and the manpower requirement.
3. The effort required to develop the project depends on what the software does.
4. FP is programming language independent.
5. FP method is used for data processing systems, business systems like information systems.
6. The five parameters mentioned above are also known as information domain characteristics.
7. All the parameters mentioned above are assigned some weights that have been experimentally determined and are shown in Table

Measurement Parameter	Low	Average	High
1.Number of external inputs (EI)	7	10	15
2. Number of external outputs (EO)	5	7	10
3. Number of external inquiries (EQ)	3	4	6
4. Number of internal files (ILF)	4	5	7
5. Number of external interfaces (EIF)	3	4	6

-- The functional complexities are multiplied with the corresponding weights against each function, and the values are added up to determine the UFP (Unadjusted Function Point) of the subsystem.

Computing FPs

Measurement Parameter	Count		Weighing factor			
			Simple	Average	Complex	
1. Number of external inputs (EI)	—	*	3	4	6 =	—
2. Number of external Output (EO)	—	*	4	5	7 =	—
3. Number of external Inquiries (EQ)	—	*	3	4	6 =	—
4. Number of internal Files (ILF)	—	*	7	10	15 =	—
5. Number of external interfaces(EIF)	—	*	5	7	10 =	—
Count-total →						

- Here that weighing factor will be simple, average, or complex for a measurement parameter type.
- The Function Point (FP) is thus calculated with the following formula.
- **FP = Count-total * [0.65 + 0.01 * ∑(f_i)]**
  = Count-total * CAF
- where Count-total is obtained from the above Table.
- **CAF = [0.65 + 0.01 *∑(f_i)]**
- and **∑(f_i)** is the sum of all 14 questionnaires and show the complexity adjustment value/ factor-CAF (where i ranges from 1 to 14). Usually, a student is provided with the value of ∑(f_i)
- Also note that **∑(f_i)** ranges from 0 to 70, i.e.,
- 0 <= ∑(f_i) <=70
- and CAF ranges from 0.65 to 1.35 because
- When **∑(f_i)** = 0 then CAF = 0.65
- When **∑(f_i)** = 70 then CAF = 0.65 + (0.01 * 70) = 0.65 + 0.7 = 1.35

- Based on the FP measure of software many other metrics can be computed:
- Errors/FP
- $/FP.
- Defects/FP
- Pages of documentation/FP
- Errors/PM.
- Productivity = FP/PM (effort is measured in person-months).
- $/Page of Documentation.

Example 2: Compute the function point, productivity, documentation, cost per function for the following data:

- Number of user inputs = 24
- Number of user outputs = 46
- Number of inquiries = 8
- Number of files = 4
- Number of external interfaces = 2
- Effort = 36.9 p-m
- Technical documents = 265 pages
- User documents = 122 pages
- Cost = $7744/ month
- Various processing complexity factors are: 4, 1, 0, 3, 3, 5, 4, 4, 3, 3, 2, 2, 4, 5.

Measurement Parameter	Count		Weighing factor
1. Number of external inputs (EI)	24	*	4 = 96
2. Number of external outputs (EO)	46	*	4 = 184
3. Number of external inquiries (EQ)	8	*	6 = 48
4. Number of internal files (ILF)	4	*	10 = 40
5. Number of external interfaces (EIF) Count-total →	2	*	5 = 10 378

So sum of all f_i ($i \leftarrow 1$ to 14) = 4 + 1 + 0 + 3 + 5 + 4 + 4 + 3 + 3 + 2 + 2 + 4 + 5 = 43

FP = Count-total * [0.65 + 0.01 *$\sum(f_i)$]
= 378 * [0.65 + 0.01 * 43]
= 378 * [0.65 + 0.43]
= 378 * 1.08 = 408

$$\text{Productivity} = \frac{\text{FP}}{\text{Effort}} = \frac{408}{36.9} = 11.1$$

Total pages of documentation = technical document + user document
= 265 + 122 = 387pages

Documentation = Pages of documentation/FP
= 387/408 = 0.94

$$\text{Cost per function} = \frac{\text{cost}}{\text{productivity}} = \frac{7744}{11.1} = \$700$$

Engineering and Production stages

- Life cycle phases consist of various separated modules with defined functionalities.
- Life cycle phases are mainly divided into two broad categories:
  1. ***Engineering Stage/Phase***
  2. ***Production Stage/Phase***

1. Engineering Phase:

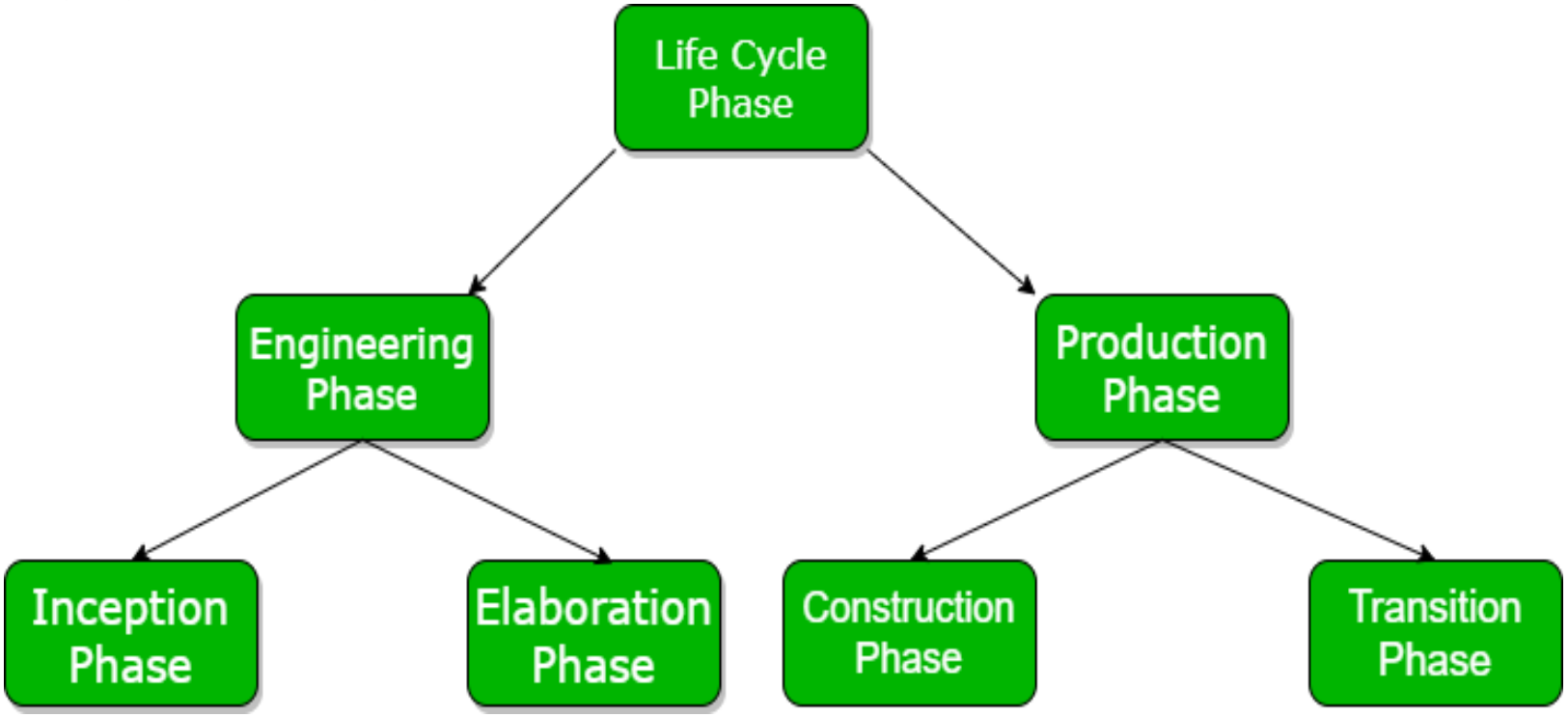
- Engineering phase involves establishing the goal and defines the overall scope of the project.
- With small team size and less predicted.
- Engineering phase is further divided into 2 Phases:
  - **Inception Phase**
  - **Elaboration Phase.**

i) Inception phase :

- involves establishing goals and gathering the requirements needed for the software development.
- It involves the cost estimation and identifying the risk factors.
- In the inception phase, we mainly work on the scope of the project and architecture.
- Feasibility analysis is also an important part of the inception phase.

ii) Elaboration Phase :

- Elaboration phase involves in-depth evaluation and study as well as establishing the strong architecture and infrastructure.
- In this phase, we work on the efficiency of our architecture and also analyze use cases and other software diagrams.
- Focus on reduce the risk to a certain extent and a preliminary user module is prepared in this phase.



2. Production Phase:

- In the Production phase, we mainly focus on the Implementation of Project and optimization including the reduced cost and risk factors of our project.
- It also involves various testing for efficient deployment of the project.
- It involves the large team size and most of the time it is predictable.
- It is broadly divided into 2 Phases:
  - **Construction Phase**
  - **Transition Phase.**

i) Construction Phase –

- In the construction phase, we perform the implementation of our software.
- In this phase, we minimize the risk and eliminate it.
- All the features and components are integrated into an application.
- In this phase, perform strict testing and process optimization is done.
- Minimize the development cost and work to improve its efficiency.
- Construction phase mainly focuses on the implementation and testing of our software.

ii) Transition Phase –

- In the Transition phase, we perform strict testing mainly beta testing and deployment of software or project.
- After receiving the feedback from the user, we perform some changes in our software to make it more efficacious.
- In this phase, the developer works on a project with an user's view to make software more supportable and user friendly.

Differences in emphasis between these two stages

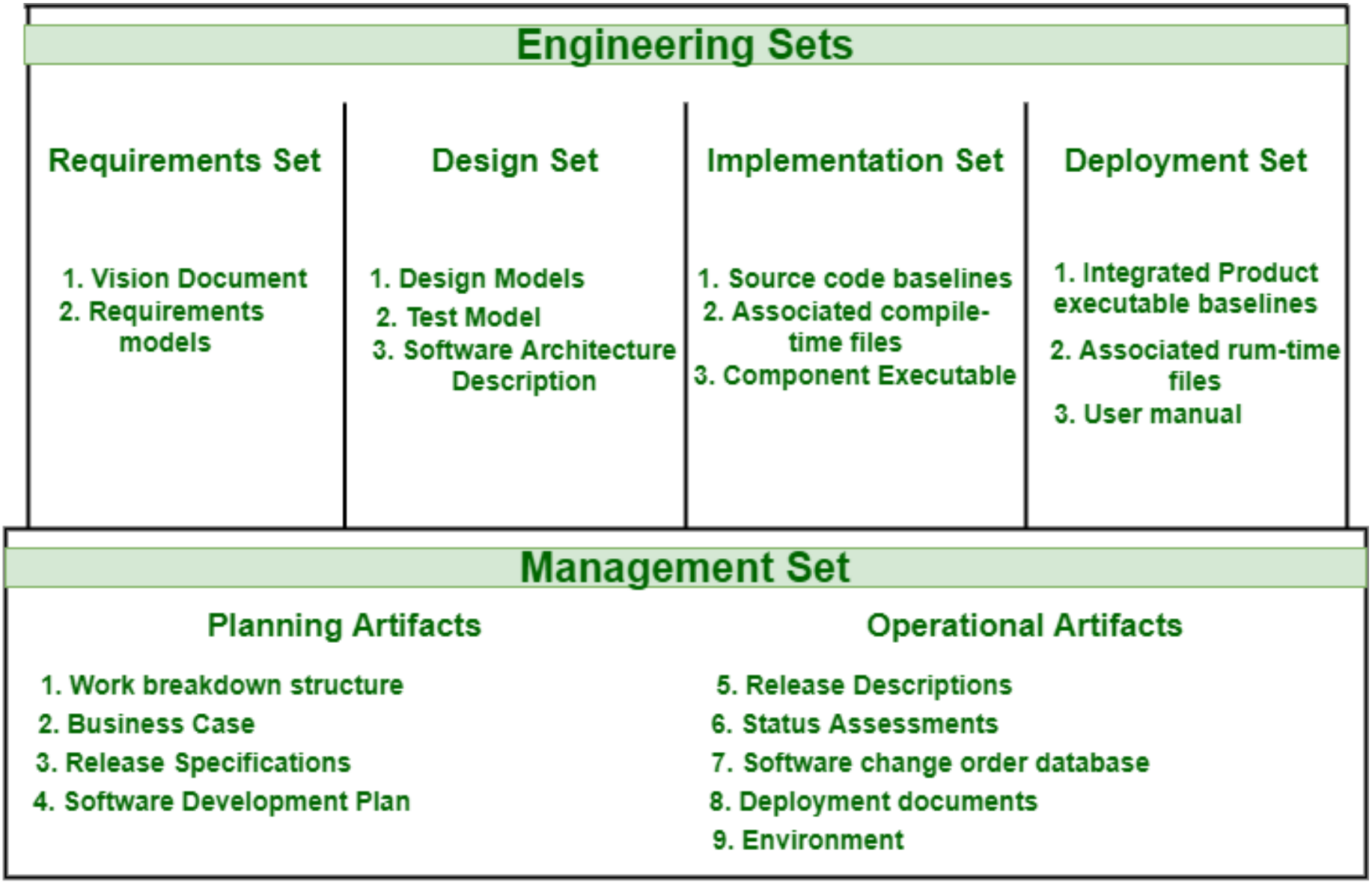
life-cycle aspect	engineering stage emphasis	production stage emphasis
Risk reduction	Schedule, technical feasibility	Cost
Products	Architecture baseline	Product release baselines
Activities	Analysis, design, planning	Implementation, testing
Assessment	Demonstration, inspection, analysis	Testing
Economics	Resolving diseconomies of scale	Exploiting economies of scale
Management	Planning	Operations

Artifacts of the Process

- **Artifact** is highly specific methods or processes of **development.**
- Processes can be project plans, business cases, or risk assessments.
- Distinct gathering and collections of detailed information are generally organized and incorporated into artifact sets.
- A set generally represents complete aspect of system.
- This is simply done to make development and establishment of complete software system in manageable manner.
- Artifacts of the life-cycle of software are generally organized and divided into two sets i.e.,

  -1. Management Artifacts (set)

  -2. Engineering Artifacts (set)



Overview of the Artifact Sets

1. **Management Set Artifacts:**
   - This set usually captures artifacts that are simply associated with planning and execution or running process.
   - These artifacts generally make use of ad hoc notations.
   - It includes text, graphics or whatever representation is required or needed to simply capture "contracts" among all project personnel (such as project developers, project management, etc.)
   - Among different stakeholders (such as user, project manager, etc.), and even between stakeholders and project personnel.
   - This set includes various artifacts such as work breakdown structure, business case, software development plan, deployment, Environment

- Artifacts of this set are evaluated, checked, and measured through combination of following :

- ***Review of relevant stakeholder.***
- ***Analyzing alterations or changes among present version of artifact and previous versions.***
- ***Demonstrations of major milestone regarding balance between all artifacts and, in specific or particular, accuracy of business case and vision artifacts.***

2. **Engineering Sets Artifacts:**

- In this set, primary mechanism or method for forming an idea regarding evolving quality of these artifact sets in transitioning of information from one set to another.
- This set is further divided into four distinct sets that include requirements set, design set, implementation set, and deployment set.

Requirement Set –

- Artifacts of this set are evaluated, checked, and measured through present vision and requirements models.
- Analysis of consistency with supplementary specifications of management set.
- Analysis of consistency among requirement models.

Design Set –

- Design model generally includes all structural and behavioral data or information .
- These set artifacts mostly include test models, design models, software architecture descriptions.

Implementation Set –

- This set generally contains source code as implementation of components, their form, interfaces, and executable that are necessary for stand-alone testing of components.

Development Set –

- Tools that are used are network management tools, test coverage, and test automation tools, etc.
- These set generally contains executable software's, build scripts, ML notations, installation scripts.

Requirements Set	Design Set	Implementation Set	Deployment Set
1. Vision document	1. Design model(s)	1. Source code	1. Integrated product
2. Requirements	2. Test model	baselines	executable
model(s)	3. Software	2. Associated	baselines
	architecture	compile-time	2. Associated

Pragmatic Artifacts

- **Pragmatic Artifacts** is the conventional document-driven approach , simply given on development, polish, format, review, update, modify and distribute documents.
- It is an approach that redirects this effort of documentation to simply improve rigor and easily understanding of source of data or information.
- With use of smart browsing and navigation tools, it also allows an on-line review of source of the native information.

This idea increases various cultural issues. Some of them are given below:

- People simply want to review data or information but sometimes don't be able to understand language of artifacts.
- People who want to review data or information don't have any access to tools required.
- Human-readable engineering artifacts must take use of rigorous notations that are fully complete, consistent, and are used in a self-documenting manner.
- Documentation that is useful is self-defining.
- Paper is tangible.

Effort Estimation Techniques: Four methods mostly used --

1) **expert opinion-based,**

2) top-down estimation,

3) bottom-up estimation

4) estimation using a parametric or algorithmic model.

The four major Cost estimation techniques used to develop cost estimates for acquisition programs are

1) **Analogy**
2) **Parametric (Statistical)**
3) **Engineering (Bottoms Up)**
4) **Actual Costs**.

COSMIC full function Points

- COSMIC function points are a unit of measure of software functional size.
- The process of measuring software size is called functional size measurement (FSM).
- COSMIC functional size measurement is applicable to business, real-time and infrastructure software at any level of decomposition.
- It is independent of the technology or processes used to develop the system.
- It is an ISO standard. The unit of size is the COSMIC Function Point or CFP.
- COSMIC is an international standard for measuring software size: ISO/IEC 19761:2011.

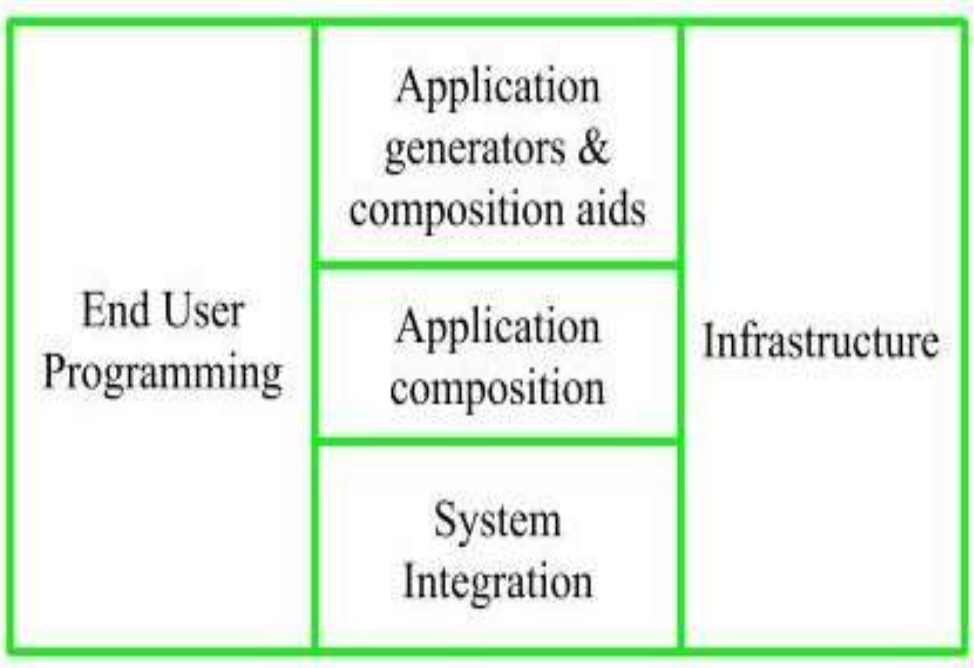
Uses

Measured (or estimated) the size in COSMIC Function Points ,base metric to :

- ***Estimate development effort***
- ***Estimate project duration***
- ***Estimate project quality achievement***
- ***Estimate test effort***
- ***Assess the value a software asset***
- ***Estimate maintenance and replacement costs***
- ***Assess the achievement of quality (defect removal rates) and more….***

COCOMO II

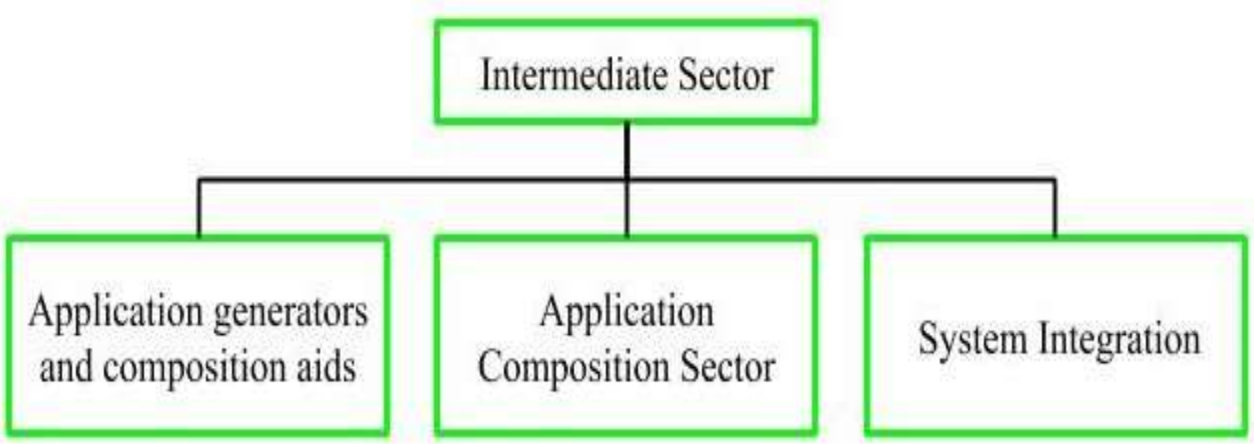
- **COCOMO-II** is the revised version of the original Cocomo (Constructive Cost Model) and is developed at University of Southern California.
- It is the model that allows one to estimate the cost, effort and schedule when planning a new software development activity.
- It consists of three sub-models:



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 this 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, domain specific components such as financial, medical or industrial process control packages.

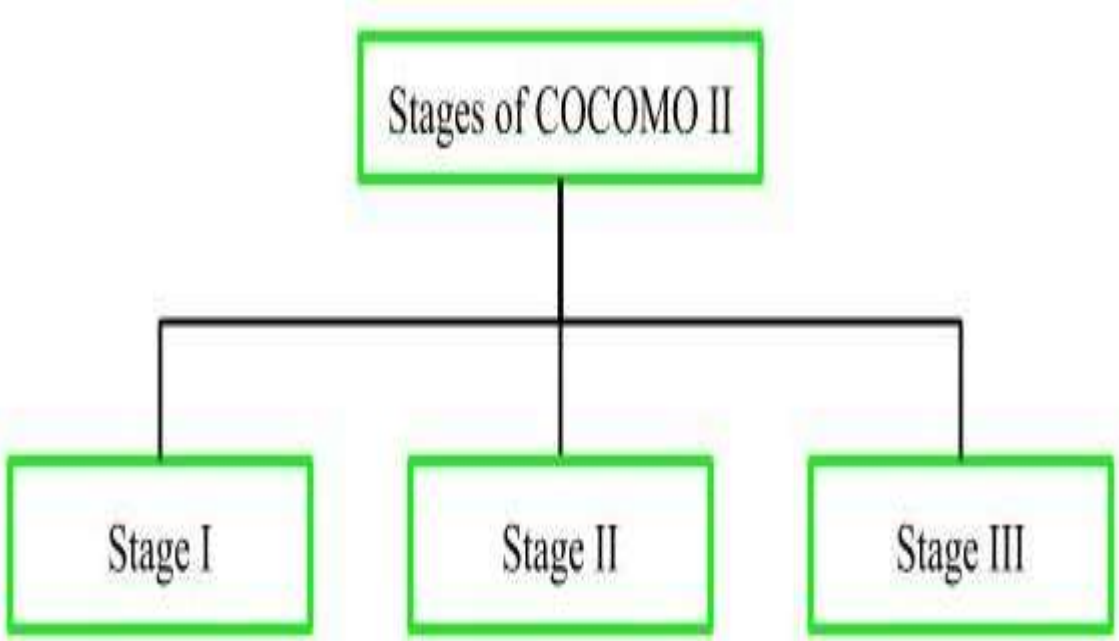
(c). System Integration –

-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:



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.

Stage-II:

- It supports estimation in the early design stage of the project, when we less know about it.
- For this it uses Early Design Estimation Model.
- This model is used in early design stage of application generators, infrastructure, system integration.

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 completion of the detailed architecture of application generator, infrastructure, system integration.

COCOMO II: A Parametric Productivity Model

- COnstructive COst MOdel II (COCOMO II) is a model that allows one to estimate the cost, effort, and schedule when planning a new software development activity.
- COCOMO II is the latest major extension to the original COCOMO (COCOMO 81) model published in 1981.
- It consists of three sub models, each one offering increased fidelity the further along one is in the project planning and design process.
- Listed in increasing fidelity, these sub models are called the Applications Composition, Early Design, and Post-architecture models

COCOMO II can be used for the following major decision situations

- Making investment or other financial decisions involving a software development effort
- Setting project budgets and schedules as a basis for planning and control
- Deciding on or negotiating tradeoffs among software cost, schedule, functionality, performance or quality factors
- Making software cost and schedule risk management decisions
- Deciding which parts of a software system to develop, reuse, lease, or purchase
- Making legacy software inventory decisions: what parts to modify, phase out, outsource,etc
- Setting mixed investment strategies to improve organization's software capability, via reuse, tools, process maturity, outsourcing, etc
- Deciding how to implement a process improvement strategy, such as that provided in the SEI CMM
- The original COCOMO model was first published by Dr. Barry Boehm in 1981, and reflected the software development practices of the day.

- In the ensuing decade and a half, software development techniques changed dramatically.
- These changes included a move away from mainframe overnight batch processing to desktop-based real-time turnaround.
- A greatly increased emphasis on reusing existing software and building new systems using off-the-shelf software components .
- Spending as much effort to design and manage the software development process as was once spent creating the software product.
- These changes and others began to make applying the original COCOMO model problematic.
- Organizations, the result is COCOMO II, a revised cost estimation model reflecting the changes in professional software development practice that have come about since the 1970s.
- This new, improved COCOMO is now ready to assist professional software cost estimators for many years to come