2 Project Planning

Syllabus

Tasks in Project Planning; Work Breakdown Structures (WBS), Planning Methods, Selecting Project Approach, SDLC, Software Processes and Process Models, Choice of Process Models, A Generic Project Model, Software Cost Estimation; COCOMO Model; Budgeting.

Contents

2.1 Project Planning : Meaning

- You may have a great idea for a project, but without planning, your project will remain just that - an idea. Planning is the critical step to take a project from an intangible theory to a tangible result.
- To bring a project to fruition, the project manager will need to assemble **a project plan**. The project plan describes the cost, scope, and schedule for the project. It lays out exactly what activities and tasks will be required, as well as the resources needed, from personnel to equipment to financing, and where they can be acquired. Good project planning also factors in risk and how to manage it, including contingency plans, and details a communication strategy to keep all stakeholders up to date and on board.
- **The purpose of the project planning phase is to :**
  - Establish business requirements
  - Establish cost, schedule, list of deliverables, and delivery dates
  - Establish resources plans
  - Obtain management approval and proceed to the next phase.

2.2 Tasks in Project Planning

- Planning the project typically involves the following steps :

1. **Initiation :** This step typically occurs before the project is greenlit. It usually involves putting together a business case document that explains the need for the project, followed by a feasibility study to determine the viability of the project in terms of its cost and projected benefits.
2. **Stakeholder involvement :** Identify your project sponsors and key stakeholders. To ensure the success of the project, meet with them to discuss their needs and expectations. Map out the project scope, budget, and timeline with them, and make sure to get their complete buy-in.
3. **Prioritizing goals :** A project - and a team - can only do so much. Prioritize your goals to make fulfilling them clearer and easier.
4. **Identifying deliverables :** What are the specific deliverables that you and your team are expected to produce ? You'll need to know exactly what is expected of you, as well as when (i.e., the deadlines for each output). You'll also want to define what success looks like for each deliverable and develop metrics to track and rank each one.

5. **Scheduling :** Using the information in the previous step, you'll need to map out the project's timeline.
6. **Developing a project plan :** As previously described, a project plan lays out the steps needed to bring the project to fruition. It includes all the activities and tasks required in the appropriate order and workflow. The project plan will draw from all the previous steps.
7. **Bake in contingency plans :** No project is without hiccups. Make sure you plan for any bumps in the road by assessing the risks associated with your project and putting plans in place to address them.

- Once the project has been mapped out, it will need to be presented to the stakeholders, edited if necessary, and then managed. Project plan management includes troubleshooting when issues arise, keeping the project on schedule, and moderating the budget.

2.3 Processes of Project Planning

The basic processes of project planning are :

- **Scope planning -** Specifying the in - scope requirements for the project to facilitate creating the work breakdown structure.
- **Preparation of the work breakdown structure -** Spelling out the breakdown of the project into tasks and sub - tasks.
- **Project schedule development -** Listing the entire schedule of the activities and detailing their sequence of implementation.
- **Resource planning -** Indicating who will do what work, at which time, and if any special skills are needed to accomplish the project tasks.
- **Budget planning -** Specifying the budgeted cost to be incurred at the completion of the project.
- **Procurement planning -** Focusing on vendors outside your company and subcontracting.
- **Risk management -** Planning for possible risks and considering optional contingency plans and mitigation strategies.
- **Quality planning -** Assessing quality criteria to be used for the project.
- **Communication planning -** Designing the communication strategy with all project stakeholders.

2.4 Work Breakdown Structure (WBS)

2.4.1 Meaning

- Breaking work into smaller tasks is a common productivity technique used to make the work more manageable and approachable. For projects, the **Work Breakdown Structure (WBS)** is the tool that utilizes this technique and is one of the most important project management documents. It single handedly integrates scope, cost and schedule baselines ensuring that project plans are in alignment.
- The work breakdown structure as a "deliverable oriented hierarchical decomposition of the work to be executed by the project team."

2.4.2 Levels of a WBS

- A WBS in project management takes large, complex projects and breaks down the project scope into more manageable pieces to make it easier to plan, schedule and deliver. Tiers of project deliverables and tasks are created to support the planning, execution, and monitoring of projects. There are four main levels of a WBS, which are outlined below :
  - **The Top Level :** The project title or final deliverable.
  - **Controls Account :** The main project phases and deliverables.
  - **Work Packages :** The group of tasks that lead to the controls account level.
  - **Activities :** The tasks needed to complete the work package.
- These tiers are found within all the different types of a work breakdown structure.

2.4.3 What is Included in a Work Breakdown Structure ?

A typical work breakdown structure is made up of several key components. They are as follows :

- **WBS Dictionary :** A document that defines the various elements of the WBS. It's an important component of a WBS because it allows the project participants and stakeholders to understand the phases, deliverables and work packages with more clarity.
- **Task Number and Description :** Giving each task a number makes it easy to identify them. A description will help define what the task is, which will provide direction for the team when it's time to execute it.
- **Task Owner :** The owner is the person, organization or department who oversees the task from assignment to completion and ensures that it has been properly executed.

- **Task Dependency :** Some of the tasks on the path to the final deliverable will have to wait until another task is done or started before they can begin. This is called a "task dependency" and requires linking the two dependent tasks together in order to avoid slippage later in the project.
- **Cost of Task :** Every task is going to have a cost associated with it. You'll want to note that to keep track of your budget.
- **Start, Finish and Estimated Completion of Task :** Add the start and finish dates for each task, and estimate the time you have on your schedule to execute it.
- **Task Status :** The status of the task will show whether it's assigned or not, in progress, late or complete, which helps with tracking.

2.4.4 How to Create a Work Breakdown Structure

- There are five steps to creating a work breakdown structure. These are the big steps, the bird's - eye view of a WBS, which eventually gets down to the granular level. But it's good to know the main parts of what is needed to construct a thorough WBS.

1. Define the project goals and objectives. Begin with the project charter the scope, objectives and who is participating in the project determine what it is and describe it.
2. The next level down is the project phases : Break the larger project statement of intent into a series of phases that will take it from conception to completion.
3. What are your deliverables ? List them all and note what is necessary for those deliverables to be deemed successfully delivered (sub - deliverables, work packages, resources, participants, etc.).
4. Take your deliverables from above and break them down into every single task and subtask that is necessary to deliver them. Make a list of all those tasks.
5. With the tasks now laid out, assign them to the team. Give each team member the tools, resources and authority they need to get the job done.

2.4.5 Types of WBS

- There are two types of WBS :

1) Deliverable - based and

2) Phase - based.

- The most common and preferred approach is the deliverable - based approach. The main difference between the two approaches is the **elements** identified in the first Level of the WBS.

❑ 1. Deliverable - based Work Breakdown Structure

- A deliverable - based work breakdown structure clearly demonstrates the relationship between the project deliverables (i.e., products, services or results) and the scope (i.e., work to be executed). Fig. 2.4.1 is an example of a deliverable - based WBS for building a house. Fig. 2.4.2 is an example of a phase - based WBS for the same project.

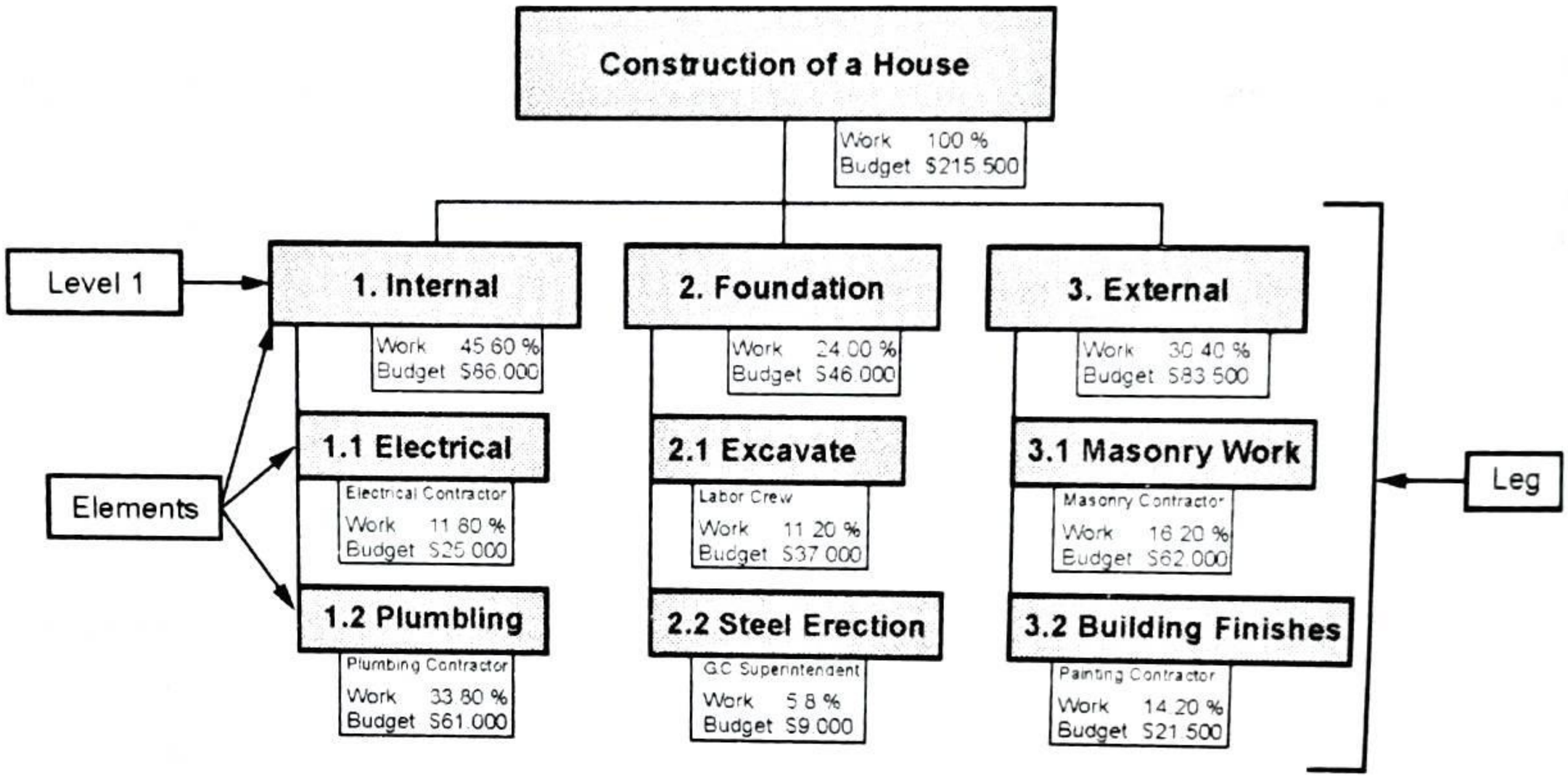

Fig. 2.4.1 : Deliverable based work breakdown structure

- In Fig. 2.4.1, the level 1 elements are summary deliverable descriptions. The level 2 elements in each leg of the WBS are all the unique deliverables required to create the respective level 1 deliverable.

❑ 2. Phase - Based Work Breakdown Structure

- In Fig. 2.4.2, a phase - based WBS, the level 1 has five elements. Each of these elements are typical phases of a project. The level 2 elements are the unique deliverables in each phase. Regardless of the type of WBS, the lower level elements are all deliverables. Notice that elements in different legs have the same name. A phase - based WBS requires work associated with multiple elements be divided into the work unique to each level 1 element. A **WBS dictionary** is created to describe the work in each element.
- A good WBS is simply one that makes the project more manageable. Every project is different; every project manager is different and every WBS is different. So, the right WBS is the one that best answers the question, "what structure makes the project more manageable ?"

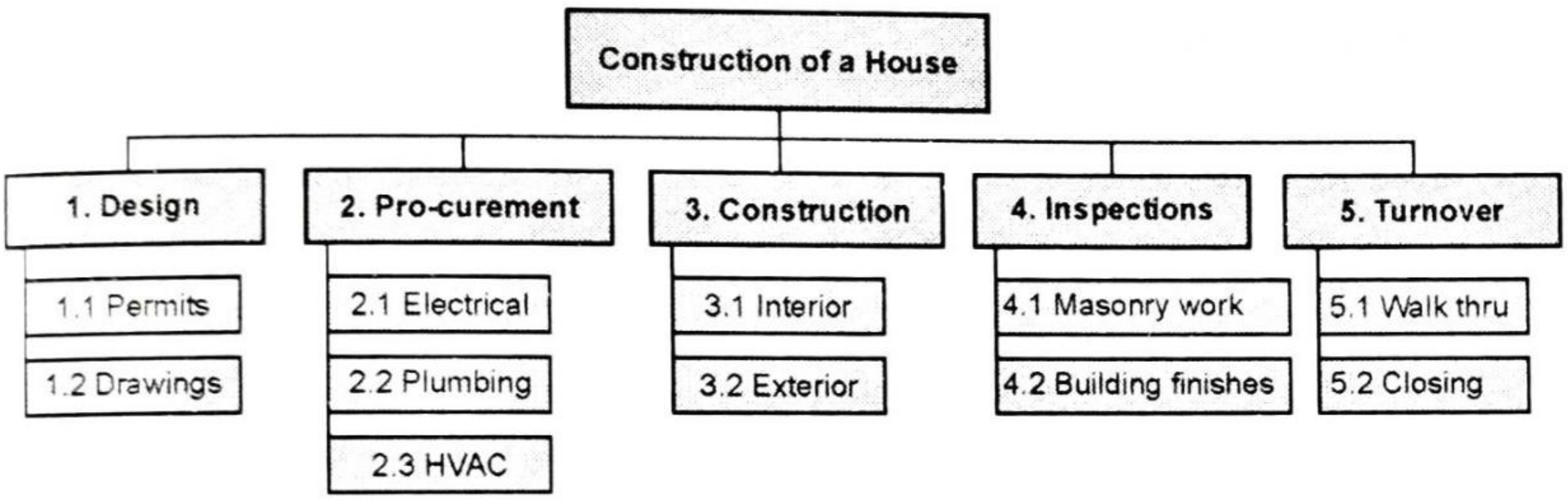

Fig. 2.4.2 : Phase based work breakdown structure

2.4.6 Benefits of WBS Software

- A work breakdown structure is an essential planning tool. Having software to assist you makes the whole process easier and quicker. It also can connect the work to your larger **project planning**, execution and more. Here's what WBS software can do for you :
  - Organize deliverables and tasks
  - Set schedule baselines
  - Visualize project schedule
  - List subtasks and dependencies
  - Estimate each task duration
  - Identify project phases.

2.5 Planning Methods

Following are the most frequently used project management methodologies in the project management practice :

1. Adaptive Project Framework

- In this methodology, the project scope is a variable. Additionally, the time and the cost are constants for the project. Therefore, during the project execution, the project scope is adjusted in order to get the maximum business value from the project.

2. Agile Software Development

- Agile software development methodology is for a project that needs extreme agility in requirements. The key features of agile are its short - termed delivery cycles (sprints), agile requirements, dynamic team culture, less restrictive project control and emphasis on real - time communication.

❑ 3. Crystal Methods

- In crystal method, the project processes are given a low priority. Instead of the processes, this method focuses more on team communication, team member skills, people and interaction. Crystal methods come under agile category.

❑ 4. Dynamic Systems Development Model (DSDM)

- This is the successor of Rapid Application Development (RAD) methodology. This is also a subset of agile software development methodology and boasts about the training and documents support this methodology has. This method emphasizes more on the active user involvement during the project life cycle.

❑ 5. Extreme Programming (XP)

- Lowering the cost of requirement changes is the main objective of extreme programming. XP emphasizes on fine scale feedback, continuous process, shared understanding and programmer welfare. In XP, there is no detailed requirements specification or software architecture built.

❑ 6. Feature Driven Development (FDD)

- This methodology is more focused on simple and well - defined processes, short iterative and feature driven delivery cycles. All the planning and execution in this project type take place based on the features.

❑ 7. Information Technology Infrastructure Library (ITIL)

- This methodology is a collection of best practices in project management. ITIL covers a broad aspect of project management which starts from the organizational management level.

❑ 8. Joint Application Development (JAD)

- Involving the client from the early stages with the project tasks is emphasized by this methodology. The project team and the client hold JAD sessions collaboratively in order to get the contribution from the client. These JAD sessions take place during the entire project life cycle.

❑ 9. Lean Development (LD)

- Lean development focuses on developing change - tolerance software. In this method, satisfying the customer comes as the highest priority. The team is motivated to provide the highest value for the money paid by the customer.

❑ 10. PRINCE2

- PRINCE2 takes a process - based approach to project management. This methodology is based on eight high - level processes.

❑ 11. Rapid Application Development (RAD)

- This methodology focuses on developing products faster with higher quality. When it comes to gathering requirements, it uses the workshop method. Prototyping is used for getting clear requirements and re - use the software components to accelerate the development timelines.
- In this method, all types of internal communications are considered informal.

❑ 12. Rational Unified Process (RUP)

- RUP tries to capture all the positive aspects of modern software development methodologies and offer them in one package. This is one of the first project management methodologies that suggested an iterative approach to software development.

❑ 13. Scrum

- This is an agile methodology. The main goal of this methodology is to improve team productivity dramatically by removing every possible burden. Scrum projects are managed by a scrum master.

❑ 14. Spiral

- Spiral methodology is the extended waterfall model with prototyping. This method is used instead of using the waterfall model for large projects.

❑ 15. Systems Development Life Cycle (SDLC)

- This is a conceptual model used in software development projects. In this method, there is a possibility of combining two or more project management methodologies for the best outcome. SDLC also heavily emphasizes on the use of documentation and has strict guidelines on it.

❑ 16. Waterfall (Traditional)

- This is the legacy model for software development projects. This methodology has been in practice for decades before the new methodologies were introduced. In this model, development life cycle has fixed phases and linear timelines. This model is not capable of addressing the challenges in the modern software development domain.

2.6 Selecting Project Approach

- It is concerned with choosing the right approach to a particular project : Variously called technical planning, project analysis, methods engineering and methods tailoring
- **In - house :** Means that the developers and the users of the software are in the same organization.
  - Often the methods to be used dictated by organizational standards Σ.
- **Suppliers :** Means that the developers and the users of the software are in the different organization.
  - Need for tailoring as different customers have different needs.

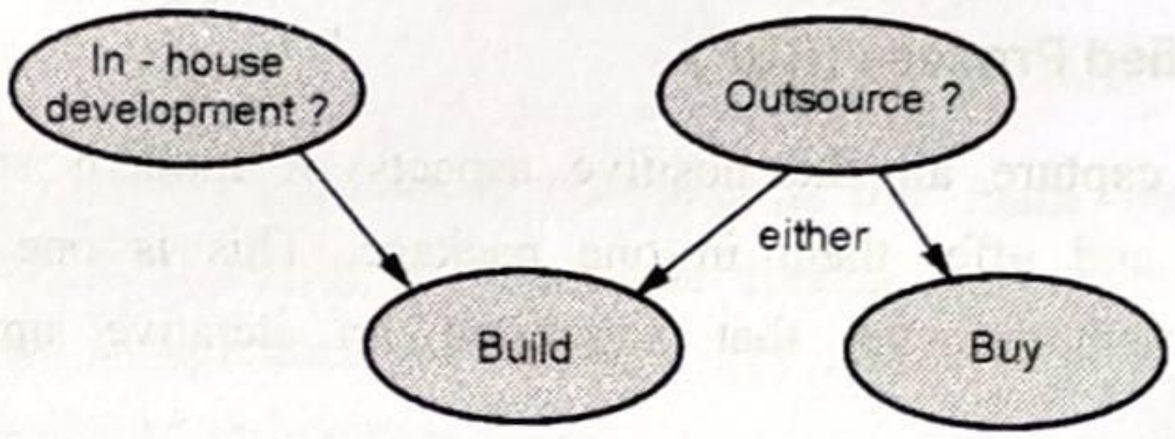

Fig. 2.6.1

❑ In - house

Developing a new IT application in - house :

- Time is needed to develop the software
- Would often require the recruitment of new technical staff to do the job
- Usually, the new staff won't be needed after the project is completed
- Sometimes due to the novelty of the project there may be lack of executives to lead the effort.

❑ Outsourcing

Contracting the project out to an external IT development company (outsourcing) :

- Time is needed to develop the software
- The conducting company will have technical and project expertise not readily available to the client
- The client would still do management effort to establish and manage the contracts.

❑ Some advantages of off - the - shelf (OTS) software

- Cheaper as supplier can spread development costs over a large number of customers

- Software already exists,
  - Can be trialled by potential customer
  - No delay while software being developed.
- Where there have been existing users, bugs are likely to have been found and eradicated.

❑ Some possible disadvantages of off-the-shelf

- Customer will have same application as everyone else : No competitive advantage, but competitive advantage may come from the way application is used.
- Customer may need to change the way they work in order to fit in with OTS application.
- Customer does not own the code and cannot change it.
- Danger of over - reliance on a single supplier.

❑ Steps of Project Analysis

- Identify project as either objective driven or product driven.
- Analyze other project characteristics by asking :
  - Will we implement a data - oriented or a process oriented system ?
  - Will the software to be produced be a general tool or application specific ?
  - Are there specific tools available for implementing the particular type of application ?
    - E.g. : Does it involve concurrent processing ?
    - Is the system knowledge - based ?
    - Will the system to be produced makes heavy use of computer graphics ?
    - Is the system to be created safety critical ?
    - Is the system designed to carry out predefined services or to be engaging and entertaining ?
    - What is the nature of the hardware/software environment in which the system will operate ?

1. Identify high - level project risks.
2. The more uncertainty in the project the more the risk that the project will be unsuccessful.
3. Recognizing the area of uncertainty allows taking steps towards reducing its uncertainty.
4. Uncertainty can be associated with the products, processes, or resources of a project.

❑ Product uncertainty

- How well are the requirements understood ?
- The users themselves could be uncertain about what the system is to do.

❑ Process uncertainty

- For the project under consideration, the organization will use an approach or an application building - tool that it never used before.

❑ Resource uncertainty

- The main area of resource uncertainty is the availability of the staff with the right ability and experience.

❑ General approach

- Look at risks and uncertainties e.g.
  - Are requirement well understood ?
  - Are technologies to be used well understood ?
- Look at the type of application being built e.g.
  - Information system ? embedded system ?
  - Criticality ? differences between target and development environments ?
- Client's own requirements
  - Need to use a particular method.

❑ Structure versus speed of delivery

Structured approach

- Also called 'heavyweight' approaches
- Step - by - step methods where each step and intermediate product is carefully defined
- Emphasis on getting quality right first time
- Example : Use of UML (Unified Modelling Language)
- Future vision : Model - Driven Architecture (MDA). UML supplemented with object constraint language, press the button and application code generated from the UML/OCL model.

Agile methods

- Emphasis on speed of delivery rather than documentation
- RAD Rapid Application Development emphasized use of quickly developed prototypes
- JAD Joint Application Development. Requirements are identified and agreed in intensive workshops with users.

Software Process Models

- Waterfall model.
- V - process model.
- Spiral model.
- Software prototyping.
- Phased development model.
  - Incremental development model.
  - Iterative development model.

2.7 Software Development Life Cycle, Software Process and Process Models

- Software Development Life Cycle (SDLC) is a framework that defines the steps involved in the development of software at each phase. It covers the detailed plan for building, deploying and maintaining the software.
- SDLC defines the complete cycle of development i.e. all the tasks involved in planning, creating, testing, and deploying a software product.

2.7.1 Software Development Life Cycle Process

- SDLC is a process that defines the various stages involved in the development of software for delivering a high - quality product. SDLC stages cover the complete life cycle of a software i.e. from inception to retirement of the product.
- Adhering to the SDLC process leads to the development of the software in a systematic and disciplined manner.

Purpose

- Purpose of SDLC is to deliver a high - quality product which is as per the customer's requirement.

- SDLC has defined its phases as, requirement gathering, designing, coding, testing, and maintenance. It is important to adhere to the phases to provide the product in a systematic manner.
- **For example**, A software has to be developed and a team is divided to work on a feature of the product and is allowed to work as they want. One of the developers decides to design first whereas the other decides to code first and the other on the documentation part.
- This will lead to project failure because of which it is necessary to have a good knowledge and understanding among the team members to deliver an expected product.

2.7.2 SDLC Cycle

- SDLC cycle represents the process of developing software.
- Below is the diagrammatic representation of the SDLC cycle :

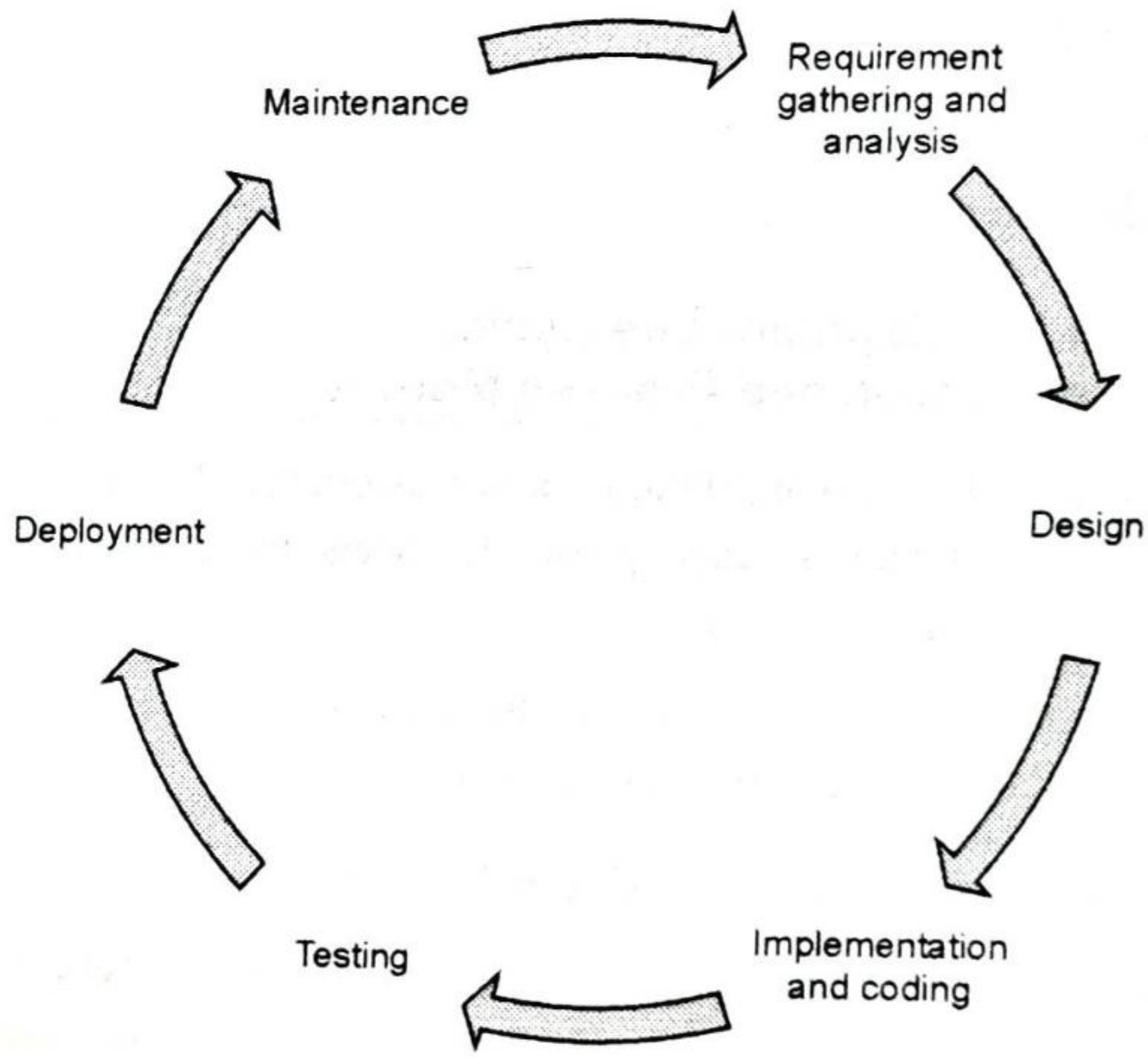

Fig. 2.7.1 : SDLC phases

- Given below are the various phases :
  - Requirement gathering and analysis
  - Design
  - Implementation or coding
  - Testing

- Deployment
- Maintenance.

1. Requirement Gathering and Analysis

- During this phase, all the relevant information is collected from the customer to develop a product as per their expectation. Any ambiguities must be resolved in this phase only.
- Business analyst and project manager set up a meeting with the customer to gather all the information like what the customer wants to build, who will be the end - user, what is the purpose of the product. Before building a product a core understanding or knowledge of the product is very important.
- **For example**, A customer wants to have an application which involves money transactions. In this case, the requirement has to be clear like what kind of transactions will be done, how it will be done, in which currency it will be done, etc.
- Once the requirement gathering is done, an analysis is done to check the feasibility of the development of a product. In case of any ambiguity, a call is set up for further discussion.
- Once the requirement is clearly understood, the SRS (Software Requirement Specification) document is created. This document should be thoroughly understood by the developers and also should be reviewed by the customer for future reference.

2. Design

- In this phase, the requirement gathered in the SRS document is used as an input and software architecture that is used for implementing system development is derived.

3. Implementation or Coding

- Implementation/Coding starts once the developer gets the design document. The software design is translated into source code. All the components of the software are implemented in this phase.

4. Testing

- Testing starts once the coding is complete and the modules are released for testing. In this phase, the developed software is tested thoroughly and any defects found are assigned to developers to get them fixed.
- Retesting, regression testing is done until the point at which the software is as per the customer's expectation. Testers refer SRS document to make sure that the software is as per the customer's standard.

5. Deployment

- Once the product is tested, it is deployed in the production environment or first UAT (User Acceptance Testing) is done depending on the customer expectation.
- In the case of UAT, a replica of the production environment is created and the customer along with the developers does the testing. If the customer finds the application as expected, then sign off is provided by the customer to go live.

6. Maintenance

- After the deployment of a product on the production environment, maintenance of the product i.e. if any issue comes up and needs to be fixed or any enhancement is to be done is taken care by the developers.

2.7.3 Software Development Life Cycle Models

A software life cycle model is a descriptive representation of the software development cycle. SDLC models might have a different approach but the basic phases and activity remain the same for all the models.

1. Waterfall Model

- **Waterfall model** is the very first model that is used in SDLC. It is also known as the linear sequential model.
- In this model, the outcome of one phase is the input for the next phase. Development of the next phase starts only when the previous phase is complete.
- First, requirement gathering and analysis is done. Once the requirement is freeze then only the system design can start. Herein, the SRS document created is the output for the requirement phase and it acts as an input for the system design.
- In system design software architecture and design, documents which act as an input for the next phase are created i.e. implementation and coding.
- In the implementation phase, coding is done and the software developed is the input for the next phase i.e. testing.
- In the testing phase, the developed code is tested thoroughly to detect the defects in the software. Defects are logged into the defect tracking tool and are retested once fixed. bug logging, retest, regression testing goes on until the time the software is in go - live state.
- In the deployment phase, the developed code is moved into production after the sign off is given by the customer.

- Any issues in the production environment are resolved by the developers which come under maintenance.

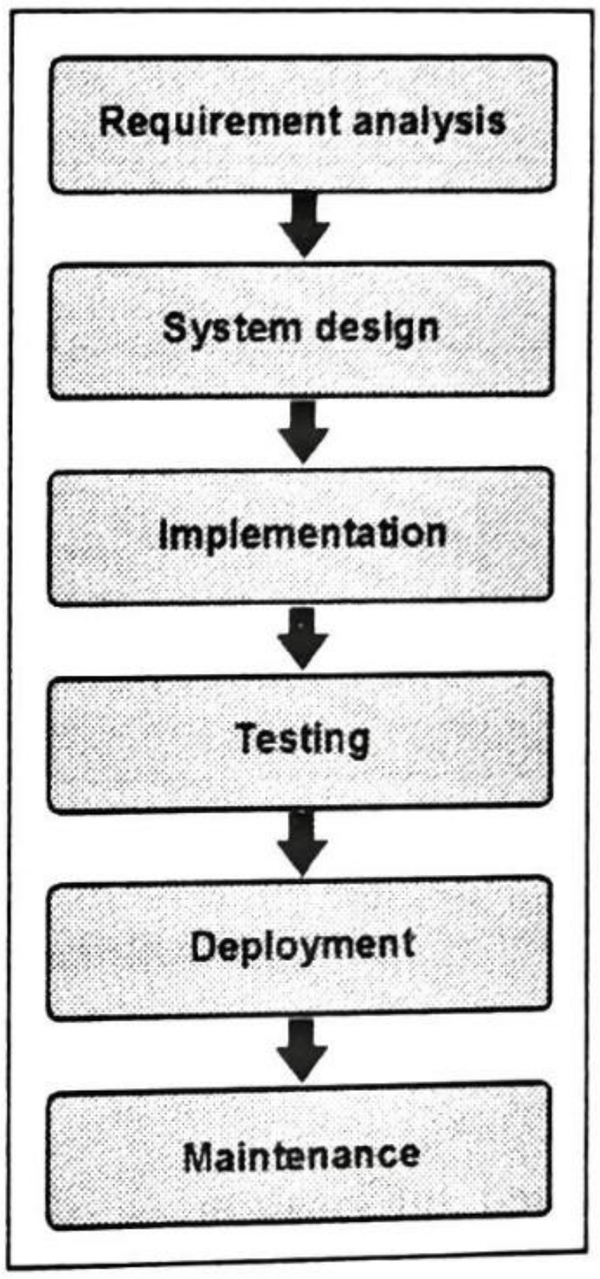

Fig. 2.7.2

Advantages of the waterfall model

- Waterfall model is the simple model which can be easily understood and is the one in which all the phases are done step by step.
- Deliverables of each phase are well defined, and this leads to no complexity and makes the project easily manageable.

Disadvantages of waterfall model

- Waterfall model is time - consuming and cannot be used in the short duration projects as in this model a new phase cannot be started until the ongoing phase is completed.
- Waterfall model cannot be used for the projects which have uncertain requirement or wherein the requirement keeps on changing as this model expects the requirement to be clear in the requirement gathering and analysis phase itself and any change in the later stages would lead to cost higher as the changes would be required in all the phases.

2. V - shaped model

- V - model is also known as **verification** and **validation** model. In this model verification and validation goes hand in hand i.e. development and testing goes parallel. V model and waterfall model are the same except that the test planning and testing start at an early stage in V - model.

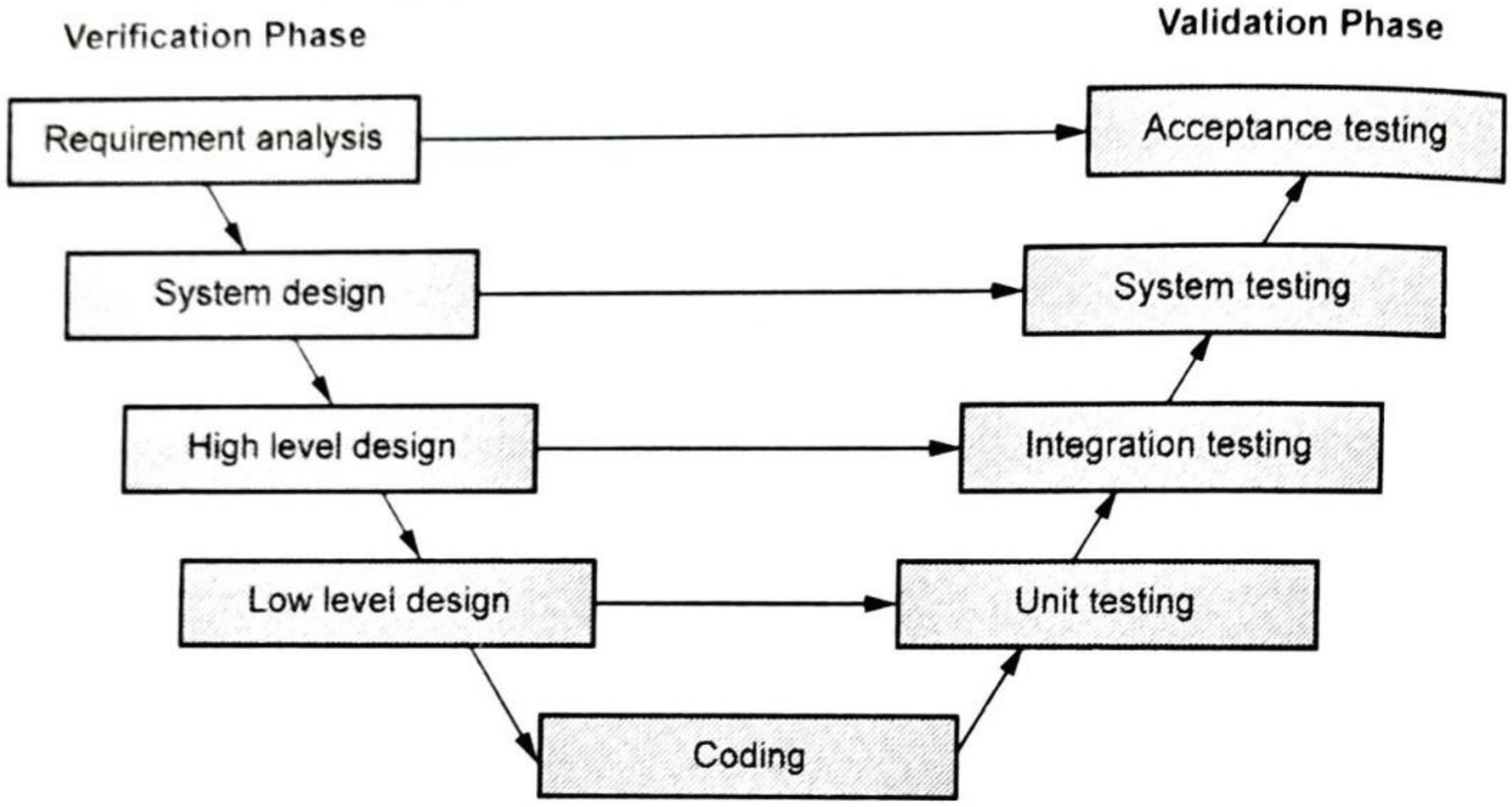

Fig. 2.7.3

a) Verification Phase :

(i) Requirement analysis :

In this phase, all the required information is gathered and analyzed. Verification activities include reviewing the requirements.

(ii) System design :

Once the requirement is clear, a system is designed i.e. architecture, components of the product are created and documented in a design document.

(iii) High - level design :

High - level design defines the architecture/design of modules. It defines the functionality between the two modules.

(iv) Low – level design :

Low - level design defines the architecture/design of individual components.

(v) Coding :

Code development is done in this phase.

b) Validation Phase :

(i) Unit testing :

Unit testing is performed using the unit test cases that are designed and is done in the low - level design phase. Unit testing is performed by the developer itself. It is performed on individual components which lead to early defect detection.

(ii) Integration testing :

Integration testing is performed using integration test cases in high - level design phase. integration testing is the testing that is done on integrated modules. It is performed by testers.

(iii) System testing :

System testing is performed in the system design phase. In this phase, the complete system is tested i.e. the entire system functionality is tested.

(iv) Acceptance testing :

Acceptance testing is associated with the requirement analysis phase and is done in the customer's environment.

Advantages of V - model

- It is a simple and easily understandable model.
- V - model approach is good for smaller projects wherein the requirement is defined and it freezes in the early stage.
- It is a systematic and disciplined model which results in a high - quality product.

Disadvantages of V - model

- V - shaped model is not good for ongoing projects.
- Requirement change at the later stage would cost too high.

3. Prototype Model

- The prototype model is a model in which the prototype is developed prior to the actual software.
- Prototype models have limited functional capabilities and inefficient performance when compared to the actual software. Dummy functions are used to create prototypes. This is a valuable mechanism for understanding the customer's needs.
- Software prototypes are built prior to the actual software to get valuable feedback from the customer. Feedbacks are implemented and the prototype is again reviewed by the customer for any change. This process goes on until the model is accepted by the customer.
- Once the requirement gathering is done, the quick design is created and the prototype which is presented to the customer for evaluation is built.

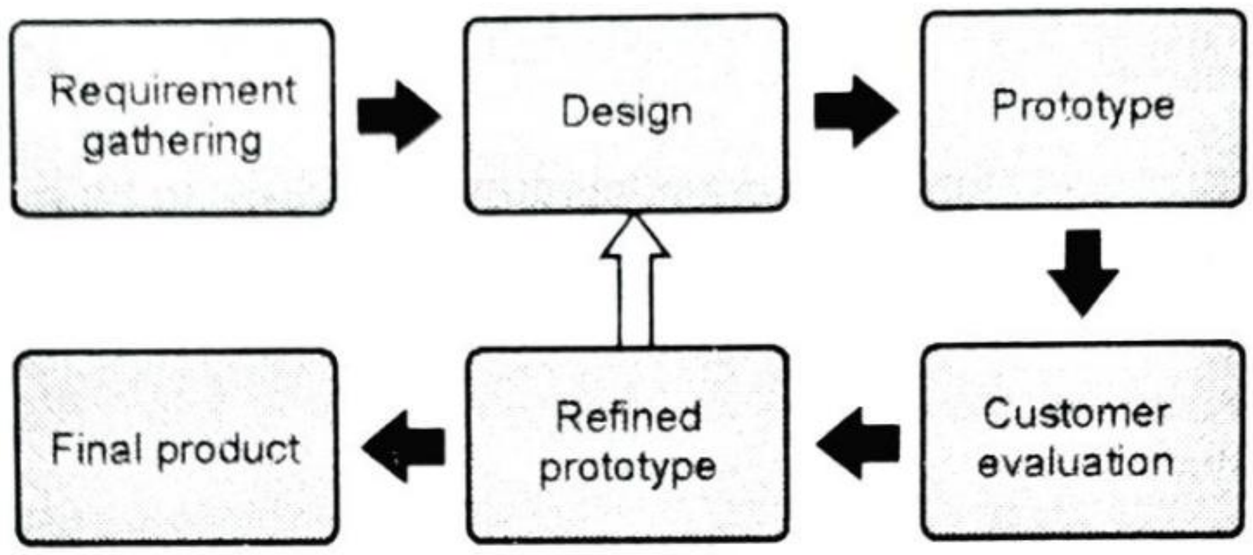

Fig. 2.7.4

- Customer feedback and the refined requirement is used to modify the prototype and is again presented to the customer for evaluation. Once the customer approves the prototype, it is used as a requirement for building the actual software. The actual software is build using the waterfall model approach.

Advantages of prototype model

- Prototype model reduces the cost and time of development as the defects are found much earlier.
- Missing feature or functionality or a change in requirement can be identified in the evaluation phase and can be implemented in the refined prototype.
- Involvement of a customer from the initial stage reduces any confusion in the requirement or understanding of any functionality.

Disadvantages of prototype model

- Since the customer is involved in every phase, the customer can change the requirement of the end product which increases the complexity of the scope and may increase the delivery time of the product.

4. Spiral Model

- The spiral model includes iterative and prototype approach. Spiral model phases are followed in the iterations. The loops in the model represent the phase of the SDLC process i.e. the innermost loop is of requirement gathering and analysis which follows the planning, Risk analysis, development, and evaluation. Next loop is designing followed by implementation and then testing.
- **Spiral Model has four phases :**
  - Planning
  - Risk Analysis
  - Engineering
  - Evaluation.

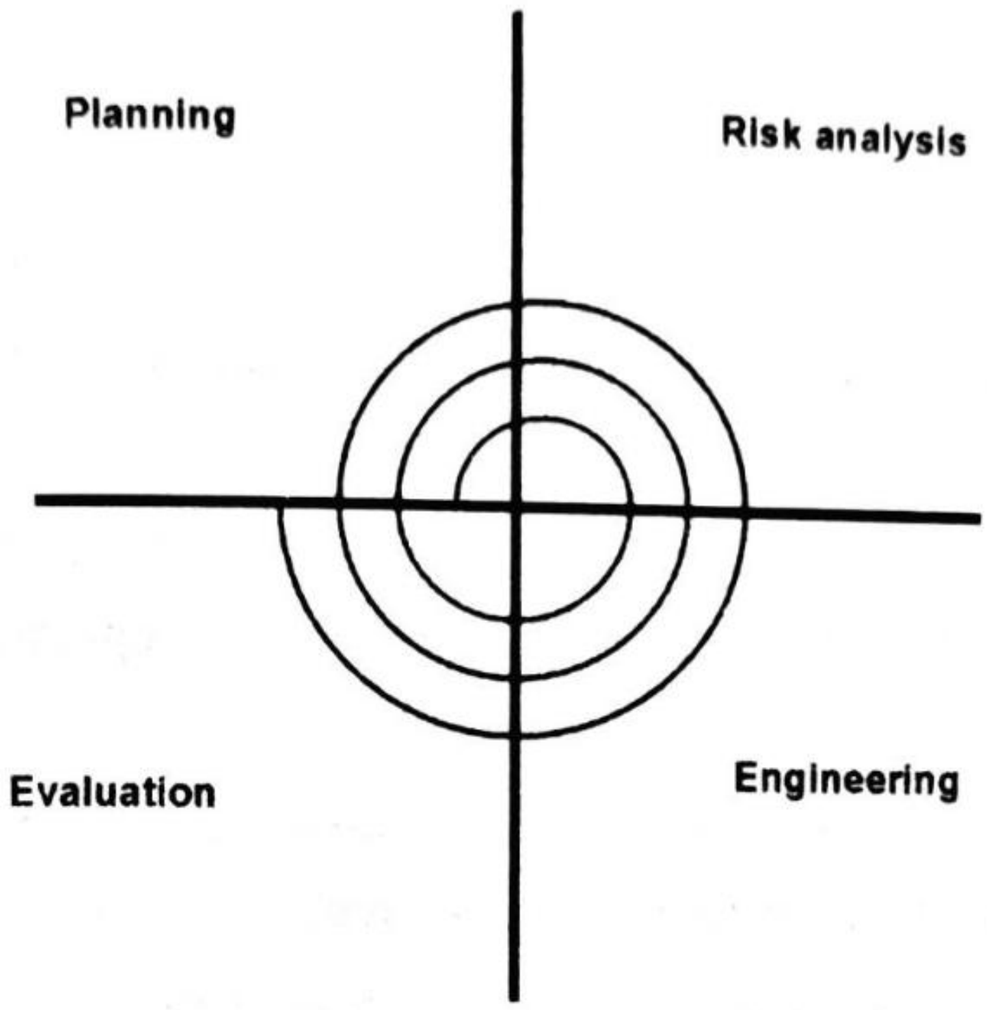

Fig. 2.7.5

(i) **Planning**

- The planning phase includes requirement gathering wherein all the required information is gathered from the customer and is documented. Software requirement specification document is created for the next phase.

(ii) **Risk Analysis**

- In this phase, the best solution is selected for the risks involved and analysis is done by building the prototype.
- **For example**, the risk involved in accessing the data from a remote database can be that the data access rate might be too slow. The risk can be resolved by building a prototype of the data access subsystem.

(iii) **Engineering**

- Once the risk analysis is done, coding and testing are done.

(iv) **Evaluation**

- Customer evaluates the developed system and plans for the next iteration.

Advantages of spiral model

- Risk analysis is done extensively using the prototype models.
- Any enhancement or change in the functionality can be done in the next iteration.

Disadvantages of spiral model

- The spiral model is best suited for large projects only.
- The cost can be high as it might take a large number of iterations which can lead to high time to reach the final product.

5. Iterative Incremental Model

- The iterative incremental model divides the product into small chunks.
- **For example**, feature to be developed in the iteration is decided and implemented. Each iteration goes through the phases namely requirement analysis, designing, coding, and testing. detailed planning is not required in iterations.
- Once the iteration is completed, a product is verified and is delivered to the customer for their evaluation and feedback. Customer's feedback is implemented in the next iteration along with the newly added feature.
- Hence, the product increments in terms of features and once the iterations are completed the final build holds all the features of the product.
- Phases of iterative and incremental development model :
  - Inception phase
  - Elaboration phase
  - Construction phase
  - Transition phase.

(i) **Inception phase :** Inception phase includes the requirement and scope of the Project.

(ii) **Elaboration phase :** In the elaboration phase, the working architecture of a product is delivered which covers the risk identified in the inception phase and also fulfills the non - functional requirements.

(iii) **Construction phase :** In the construction phase, the architecture is filled in with the code which is ready to be deployed and is created through analysis, designing, implementation, and testing of the functional requirement.

(iv) **Transition phase :** In the transition phase, the product is deployed in the production environment.

Advantages of Iterative and Incremental Model

- Any change in the requirement can be easily done and would not cost as there is a scope of incorporating the new requirement in the next iteration.
- Risk is analyzed and identified in the iterations.
- Defects are detected at an early stage.
- As the product is divided into smaller chunks it is easy to manage the product.

Disadvantages of Iterative and Incremental Model

- Complete requirement and understanding of a product are required to break down and build incrementally.

6. Big Bang Model

- Big Bang model does not have any defined process. Money and efforts are put together as the input and output come as a developed product which might be or might not be the same as what the customer needs.
- Big Bang model does not require much planning and scheduling. The developer does the requirement analysis and coding and develops the product as per his understanding. This model is used for small projects only. There is no testing team and no formal testing is done, and this could be a cause for the failure of the project.

Advantages of Big Bang model

- It's a very simple model.
- Less planning and scheduling is required.
- The developer has the flexibility to build the software of their own.

Disadvantages of the Big Bang model

- Big Bang models cannot be used for large, ongoing and complex projects.
- High risk and uncertainty.

7. Agile Model

- Agile Model is a combination of the iterative and incremental model. This model focuses more on flexibility while developing a product rather than on the requirement.
- In Agile, a product is broken into small incremental builds. It is not developed as a complete product in one go. Each build increments in terms of features. The next build is built on previous functionality.
- In agile iterations are termed as sprints. Each sprint lasts for 2 - 4 weeks. At the end of each sprint, the product owner verifies the product and after his approval, it is delivered to the customer.
- Customer feedback is taken for improvement and his suggestions and enhancement are worked on in the next sprint. Testing is done in each sprint to minimize the risk of any failures.

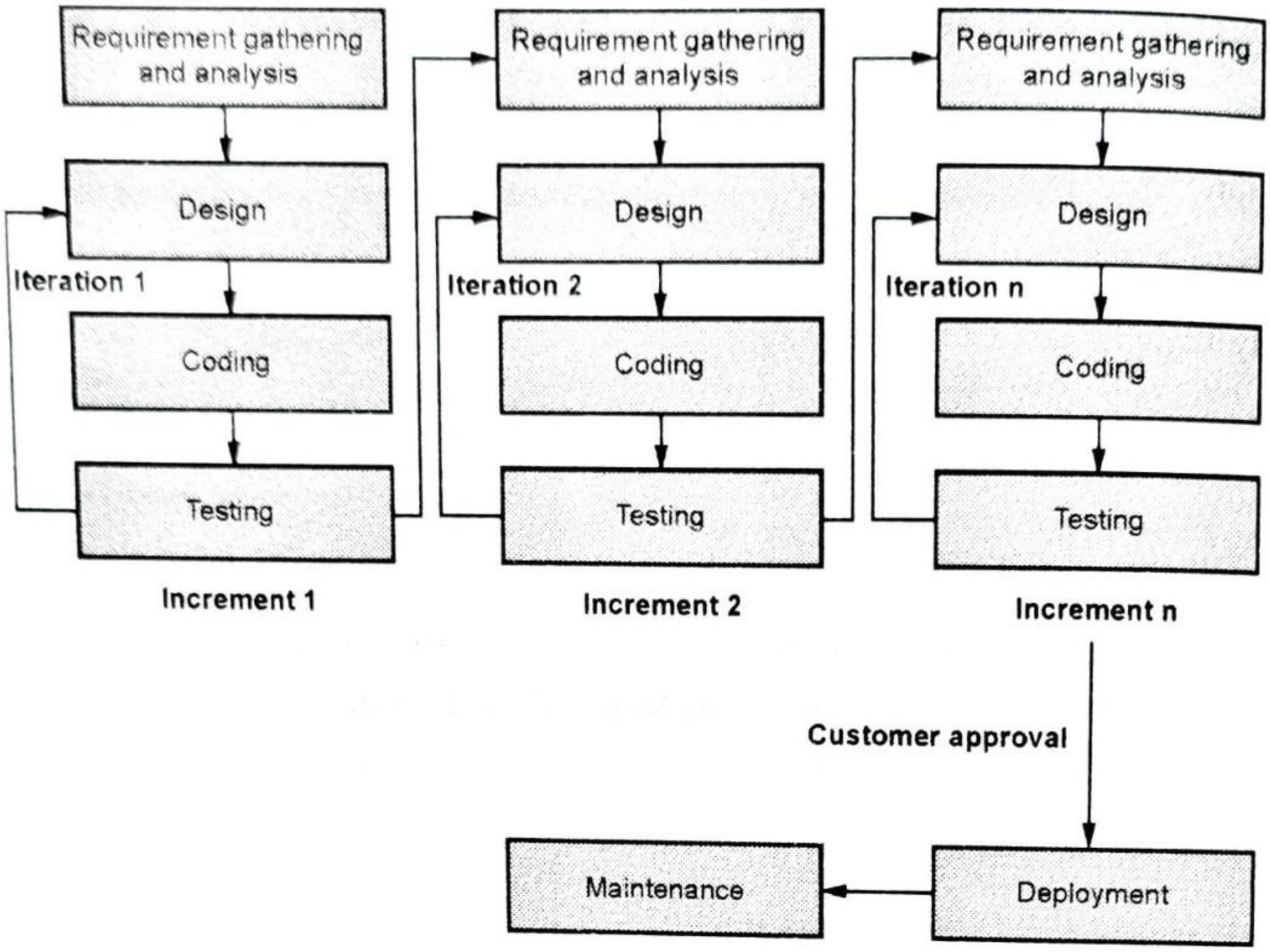

Fig. 2.7.6

Advantages of Agile model

- It allows more flexibility to adapt to the changes.
- The new feature can be added easily.
- Customer satisfaction as the feedback and suggestions are taken at every stage.

Disadvantages

- Lack of documentation.
- Agile needs experienced and highly skilled resources.
- If a customer is not clear about how exactly they want the product to be, then the project would fail.

2.8 Choice of Process Models

- Adherence to a suitable life cycle is very important, for the successful completion of the project. This, in turn, makes the management easier.
- Different software development life cycle models have their own pros and cons. The best model for any project can be determined by the factors like requirement (whether it is clear or unclear), system complexity, size of the project, cost, skill limitation, etc.

- **Example**, in case of an unclear requirement, Spiral and Agile models are best to be used as the required change can be accommodated easily at any stage.
- Waterfall model is a basic model and all the other SDLC models are based on that only.

❑ 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. You need to keep the following factors in mind when selecting your software process model :

❑ 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. Will the user need to specify requirements in detail after each iterative session ? Will the requirements *change* during the development process ?

❑ 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.

❑ 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.

❑ Cost of delay

Is the project highly time - bound with a huge cost of delay, or are the timelines flexible ?

❑ Customer involvement

Do you need to consult the customers during the process ? Does the user need to participate in all phases ?

❑ Familiarity with technology

This involves the developer's knowledge and experience with the project domain, software tools, language, and methods needed for development.

❑ Project resources

This involves the amount and availability of funds, staff, and other resources.

2.9 A Generic Project Model

- A software process is a collection of various activities.
- **There are five generic process framework activities :**

1. Communication

- The software development starts with the communication between customer and developer.

2. Planning

- It consists of complete estimation, scheduling for project development and tracking.

3. Modeling

- Modeling consists of complete requirement analysis and the design of the project like algorithm, flowchart etc.
- The algorithm is the step - by - step solution of the problem and the flow chart shows a complete flow diagram of a program.

4. Construction

- Construction consists of code generation and the testing part.
- Coding part implements the design details using an appropriate programming language.
- Testing is to check whether the flow of coding is correct or not.
- Testing also check that the program provides desired output.

5. Deployment

- Deployment step consists of delivering the product to the customer and take feedback from them.
- If the customer wants some corrections or demands for the additional capabilities, then the change is required for improvement in the quality of the software.

2.10 Software Cost Estimation

- Software development processes are split into a number of separate activities. Cost estimation of software development project focuses on, how associating estimates of effort and time with the project activities. Estimation involves answering the following questions :

1. How much effort is required to complete each activity ?
2. How much calendar time is needed to complete each activity ?
3. What is the total cost of each activity ?

- Project cost estimation and project scheduling are usually carried out together. The costs of development are primarily the costs of the effort involved, so the effort computation is used in both the cost and the schedule estimate. The initial cost estimates may be used to establish a budget for the project and to set a price for the software for a customer. The total cost of a software development project is the sum of following costs :

1. Hardware and software costs including maintenance.
2. Travel and training costs.
3. Effort costs of paying software developers.

- For most projects, the dominant cost is the effort cost. Effort costs are not just the salaries of the software engineers who are involved in the project. The following overhead costs are all part of the total effort cost :

1. Costs of heating and lighting offices.
2. Costs of support staff (accountants, administrators, system managers, cleaners, technicians etc.).
3. Costs of networking and communications.
4. Costs of central facilities (library, recreational facilities, etc.).
5. Costs of social security and employee benefits such as pensions and health insurance.

- The aim of software costing is to accurately predict the cost of developing the software. The price of software is normally the sum of development cost and profit. During the development project managers should regularly update their cost and schedule estimates. This helps with the planning process and the effective use of resources. If actual expenditure is significantly greater than the estimates, then the project manager must take some action. This may involve applying for additional resources for the project or modifying the work to be done.

1. Software productivity estimation

- Productivity estimates help to define the project cost and schedule. In a software development project managers may be faced with the problem of estimating the productivity of software engineers. For any software problem, there may be many different solutions, each of which has different attributes. One solution may execute more efficiently while another may be more readable and easier to maintain and comparing their production rates is very difficult.

- Productivity estimates are usually based on measuring attributes of the software and dividing this by the total effort required for development. Software metrics have two main types :
  1. Size - related software metrics. These metrics are related to the size of software. The most frequently used size - related metric is lines of delivered source code.
  2. Function - related software metrics. These are related to the overall functionality of the software. For example, function points and object points are metrics of this type.
- The productivity estimation defined as lines of source code per programmer month is widely used software productivity metric. It is computed by counting the total number of lines of source code divided by the total development time in programmer - months required to complete the project.
- In other cases, the total number of function points in a program measures or estimates the following program features :
  1. External inputs and outputs,
  2. User interactions,
  3. External interfaces,
  4. Files used by the system.
- Obviously, some features are more complex than others and take longer time to code. The function point metric takes this into account by application of a weighting factor. The unadjusted function point count (UFC) is computed by multiplying the number of a given feature by its estimated weight for all features and finally summing products :

$$UFC = \sum (\text{Number of a given feature}) \times (\text{Weight for the given feature})$$

- Object points are an alternative to function points. The number of object points is computed by the estimated weights of objects :
  1. Separate screens that are displayed. Simple screens count as 1 object point, moderately complex screens count as 2, and very complex screens count as 3 object points.
  2. Reports that are produced. For simple reports, count 2 object points, for moderately complex reports, count 5, and for reports that are likely to be difficult to produce, count 8 object points.
  3. Modules that are developed to supplement the database programming code. Each of these modules counts as 10 object points.

- The final code size is calculated by the number of function points and the estimated average number of lines of code (AVC) required to implement a function point. The estimated code size is computed as follows :

$$\text{CODE SIZE} = \text{AVC} \times \text{UFC}$$

- The problem with measures based on the amount produced in a given time period is that they take no account of quality characteristics such as reliability and maintainability.

2. Development cost estimation techniques

- In the early stage in a project it is very difficult to accurately estimate system development costs. There is no simple way to make an accurate estimate of the effort required to develop software. Techniques listed in Table 2.10.1 may be used to make software effort and cost estimates. All of these techniques based on the experience of project managers who use their knowledge of previous projects to estimate of the resources required for the project.

Technique	Description
Algorithmic cost modelling	Relating some software metric a mathematical model is developed to estimate the project cost.
Expert judgement	Several experts on the proposed software development techniques and the application domain are asked to estimate the project cost. The estimation process iterates until an agreed estimate is reached.
Estimation by previous projects	The cost of a new project is estimated by a completed project in the same application domain.
Application of Parkinson's law	Parkinson's law states that work expands to fill the time available and the cost is determined by the resources used.
Pricing to win	The software cost is estimated by the price what the customer has available to spend on the project.

Table 2.10.1 : Cost estimation techniques

- Each estimation technique listed in Table has its own strengths and weaknesses. It is recommended to use several cost estimation techniques and compare their results. If they predict significantly different costs more information is required about the product, the software process, development team, etc.

❑ 3. Algorithmic cost modelling

- Algorithmic cost modelling uses a mathematical expression to predict project costs based on estimates of the project size, the number of software engineers, and other process and product factors. An algorithmic cost model can be developed by analyzing the costs and attributes of completed projects and finding the closest fit mathematical expression to actual project. In general, an algorithmic cost estimate for software cost can be expressed as :

$$\text{EFFORT} = A \times \text{SIZE}\, B \times M$$

- In this equation A is a constant factor that depends on local organizational practices and the type of software that is developed. Variable SIZE may be either the code size or the functionality of software expressed in function or object points. M is a multiplier made by combining process, product and development attributes, such as the dependability requirements for the software and the experience of the development team. The exponential component B associated with the size estimate expresses the non - linearity of costs with project size. As the size of the software increases, extra costs are emerged. The value of exponent B usually lies between 1 and 1.5.
- All algorithmic models have the same difficulties :
  1. It is difficult to estimate SIZE in the early stage of development. Function or object point estimates can be produced easier than estimates of code size but are often inaccurate.
  2. The estimates of the factors contributing to B and M are subjective. Estimates vary from one person to another person, depending on their background and experience with the type of system that is being developed.
- The number of lines of source code in software is the basic software metric used in many algorithmic cost models. The code size can be estimated by previous projects, by converting function or object points to code size, by using a reference component to estimate the component size, etc. The programming language used for system development also affects the number of lines of code to be implemented. Furthermore, it may be possible to reuse codes from previous projects and the size estimate has to be adjusted to take this into account.

2.11 COCOMO Model

2.11.1 Definition and Meaning

Definition

- The COCOMO (Constructive Cost Model) is one of the most popularly **used software cost estimation** models i.e. it estimates or predicts the effort required for the project, total project cost and scheduled time for the project. This model depends on the number of lines of code for software product development. It was developed by a software engineer Barry Boehm in 1981.

Meaning

- The COCOMO estimates the cost for software product development in terms of effort (resources required to complete the project work) and schedule (time required to complete the project work) based on the size of the software product. It estimates the required number of Man-Months (MM) for the full development of software products. According to COCOMO, there are three modes of software development projects that depend on complexity. Such as :

1. Organic Project

- It belongs to small and simple software projects which are handled by a small team with good domain knowledge and few rigid requirements.
- **Example :** Small data processing or Inventory management system.

2. Semidetached Project

- It is an intermediate (in terms of size and complexity) project, where the team having mixed experience (both experience and inexperience resources) to deals with rigid/nonrigid requirements.
- **Example :** Database design or OS development.

3. Embedded Project

- This project having a high level of complexity with a large team size by considering all sets of parameters (software, hardware and operational).
- **Example :** Banking software or traffic light control software.

2.11.2 Types of COCOMO Model

Depending upon the complexity of the project the COCOMO has three types. Such as :

1. The Basic COCOMO

- It is the one type of static model to estimates software development effort quickly and roughly. It mainly deals with the number of lines of code and the level of estimation accuracy is less as we don't consider the all parameters belongs to the project. The estimated effort and scheduled time for the project are given by the relation :

$$\text{Effort (E)} = a*(\text{KLOC})^{b} \text{ MM}$$

$$\text{Scheduled Time (D)} = c*(E)^{d} \text{ Months (M)}$$

Where,

- E = Total effort required for the project in Man - Months (MM).
- D = Total time required for project development in Months (M).
- KLOC = The size of the code for the project in Kilo lines of code.
- a, b, c, d = The constant parameters for a software project.

Project type	a	b	c	d
Organic	2.4	1.05	2.5	0.38
Semidetached	3	1.12	2.5	0.35
Embedded	3.6	1.2	2.5	0.32

Example 2.11.1 : *For a given project was estimated with a size of 300 KLOC. Calculate the effort, scheduled time for development. Also, calculate the average resource size and productivity of the software for organic project type.*

Solution : Given estimated size of project is : 300 KLOC

For Organic,

$$\text{Effort (E)} = a*(\text{KLOC})^{b}$$

$$= 2.4*(300)^{1.05} = 957.61 \text{ MM}$$

$$\text{Scheduled Time (D)} = c*(E)^{d}$$

$$= 2.5*(957.61)^{0.38} = 33.95 \text{ Months (M)}$$

$$\text{Avg. Resource Size} = E/D = 957.61/33.95 = 28.21 \text{ Mans}$$

$$\text{Productivity of Software} = \text{KLOC}/E = 300/957.61 = 0.3132 \text{ KLOC/MM} = 313 \text{ LOC/MM}$$

For Semidetached

$$\text{Effort (E)} = a*(KLOC)^{b} = 3.0*(300)^{1.12} = 1784.42 \text{ MM}$$

$$\text{Scheduled Time (D)} = c*(E)^{d} = 2.5*(1784.42)^{0.35} = 34.35 \text{ Months (M)}$$

For Embedded

$$\text{Effort (E)} = a*(KLOC)^{b} = 3.6*(300)^{1.2} = 3379.46 \text{ MM}$$

$$\text{Scheduled Time (D)} = c*(E)^{d} = 2.5*(3379.46)^{0.32} = 33.66 \text{ Months (M)}$$

❑ 2. The Intermediate COCOMO

- The intermediate model estimates software development effort in terms of size of the program and other related cost drivers parameters (product parameter, hardware parameter, resource parameter, and project parameter) of the project. The estimated effort and scheduled time are given by the relationship :

$$\text{Effort (E)} = a*(KLOC)^{b} * \text{EAF MM}$$

$$\text{Scheduled Time (D)} = c*(E)^{d} \text{ Months (M)}$$

Where,

- E = Total effort required for the project in Man - Months (MM).
- D = Total time required for project development in Months (M).
- KLOC = The size of the code for the project in Kilo lines of code.
- a, b, c, d = The constant parameters for the software project.

- EAF = It is an effort adjustment factor, which is calculated by multiplying the parameter values of different cost driver parameters. For ideal, the value is 1.

Cost drivers parameters	Very low	low	Nominal	High	Very high
Product Parameter					
Required Software	0.75	0.88	1	1.15	1.4
Size of Project Database	NA	0.94		1.08	1.16
Complexity of The Project	0.7	0.85		1.15	1.3
Hardware Parameter					
Performance Restriction	NA	NA	1	1.11	1.3
Memory Restriction	NA	NA		1.06	1.21
Virtual Machine Environment	NA	0.87		1.15	1.3
Required Turnabout Time	NA	0.94		1.07	1.15

Cost drivers parameters	Very low	low	Nominal	High	Very high
Personnel Parameter					
Analysis Capability	1.46	1.19	1	0.86	0.71
Application Experience	1.29	1.13		0.91	0.82
Software Engineer Capability	1.42	1.17		0.86	0.7
Virtual Machine Experience	1.21	1.1		0.9	NA
Programming Experience	1.14	1.07		0.95	NA
Project Parameter					
Software Engineering Methods	1.24	1.1	1	0.91	0.82
Use of Software Tools	1.24	1.1		0.91	0.83
Development Time	1.23	1.08		1.04	1.1

Example 2.11.2 : *For a given project was estimated with a size of 300 KLOC. Calculate the effort, scheduled time for development by considering developer having high application experience and very low experience in programming.*

Solution : Given the estimated size of the project is : 300 KLOC

Developer having highly application experience : 0.82 (as per above table)

Developer having very low experience in programming : 1.14 (as per above table)

$$EAF = 0.82*1.14 = 0.9348$$

$$\text{Effort (E)} = a*(KLOC)^{b} * EAF = 3.0*(300)^{1.12} * 0.9348 = 1668.07 \text{ MM}$$

$$\text{Scheduled Time (D)} = c*(E)^{d} = 2.5*(1668.07)^{0.35} = 33.55 \text{ Months (M)}$$

3. The Detailed COCOMO

- It is the advanced model that estimates the software development effort like Intermediate COCOMO in each stage of the software development life cycle process.

2.11.3 Advantages and Disadvantages of COCOMO Model

Following are some advantages and disadvantages of the COCOMO model.

Advantages

- Easy to estimate the total cost of the project.
- Easy to implement with various factors.
- Provide ideas about historical projects.

Disadvantages

- It ignores requirements, customer skills, and hardware issues.
- It limits the accuracy of the software costs.
- It mostly depends on time factors.

2.12 Budgeting

- Cost budgeting can be viewed as part of estimation or as its own separate process.
- Budgeting is the process of allocating costs to a certain chunk of the project, such as individual tasks or modules, for a specific time period.
- Budgets include contingency reserves allocated to manage unexpected costs.
- For example, let's say the total costs estimated for a project that runs over three years is $2 million. However, since the budget allocation is a function of time, the project manager decides to consider just the first two quarters for now. They identify the work items to be completed and allocate a budget of, say, $35,000 for this time period, and these work items.
- The project manager uses the WBS and some of the estimation methods.
- Budgeting creates a cost baseline against which we can continue to measure and evaluate the project cost performance. If not for the budget, the total estimated cost would remain an abstract figure and it would be difficult to measure midway.
- Evaluation of project performance gives an opportunity to assess how much budget needs to be released for future phases of the project.
- Another reason to firm up budgets is that organizations often rely on expected future cash flows for their funding. During the initial phases, the project manager has a limited financial pool and has to set targets accordingly.
- It's similar to building the foundation and one floor of the house in the initial few months and later completing the rest of the project, as you save more.

❑❑❑