Syllabus: MODULE 1:- PHASES OF SOFTWARE DEVELOPMENT

- ***1. To Understand Phases and Life cycle models of Software Development***
- *1. Define software engineering and its importance 2. Explain emergence of software engineering 3. Describe Software Process 4. State Phases of software development 5. Describe Feasibility study 6. Describe Requirement Analysis 7. Describe Design phase 8. Describe Implementation phase 9. Describe testing phase 10. Describe Maintenance phase 11. Describe Life Cycle Models- Classical waterfall, Iterative, prototyping, Spiral and Agile*

1. Define Software Engineering *and its importance*

- Software engineering discusses systematic and cost-effective techniques to software development.
- These techniques have resulted from innovations as well as lessons learnt from past mistakes.
- Alternatively, we can view software engineering as the engineering approach to develop software.
- In short software engineering as a discipline provides tools and techniques for developing the software in an orderly fashion.
- The advantages of using software engineering for developing software are:
  - Improved requirement specification
  - Improved quality
  - Improved cost and schedule estimates
  - Better use of automated tools and techniques
  - Less defects in final product
  - Better maintenance of delivered software
  - Well defined processes
  - Improved productivity
  - Improved reliability

Evolution and Impact

- The early programmers used an exploratory also called build and fix programming style.

- In the build and fix (exploratory) style, normally a poor quality program is quickly developed without making any specification, plan or design. The different imperfections that are subsequently noticed while using or testing are fixed.
- The exploratory programming style is very informal style of program approach, and there are no set of rules or recommendations that one has to obey, it is guided by his own intuitions, experience, and fancies.
- The exploratory style is also the one that is normally adopted by students and novice programmers who do not have any exposure to software engineering principles.
- We can consider the exploratory program development style as an art since, any art is mostly guided by the intuition.
- The following diagram shows evolution of technology with time.

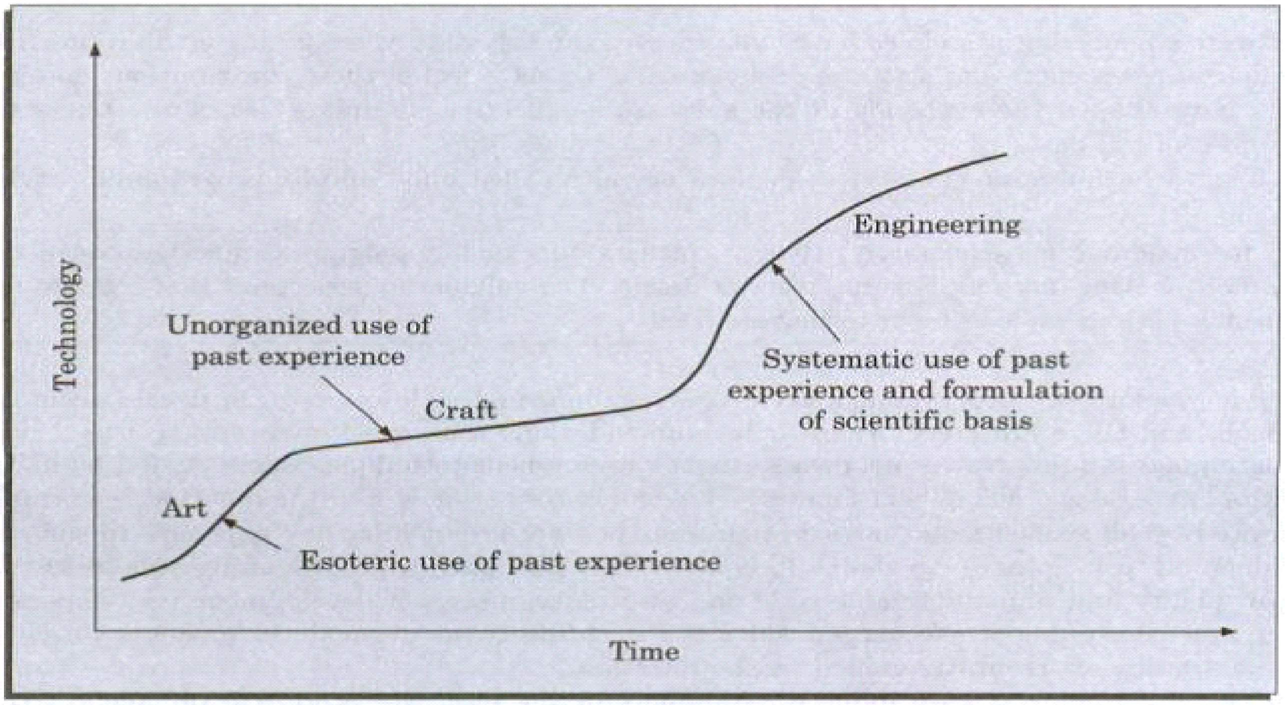

Technology
Engineering
Unorganized use of past experience
Craft
Systematic use of past experience and formulation of scientific basis
Art
Esoteric use of past experience
Time

Figure 1.1: Evolution of technology with time.

- If we analyze the evolution software development styles over the last fifty years, we can easily notice that it has evolved from an esoteric art form to a craft form, and then has slowly emerged as an engineering discipline.
- Every technology initially starts as a form of art. Overtime, it graduates to a craft, and finally emerges as an engineering principle.

- Software engineering principles are being widely used in industries and new principles are emerging.

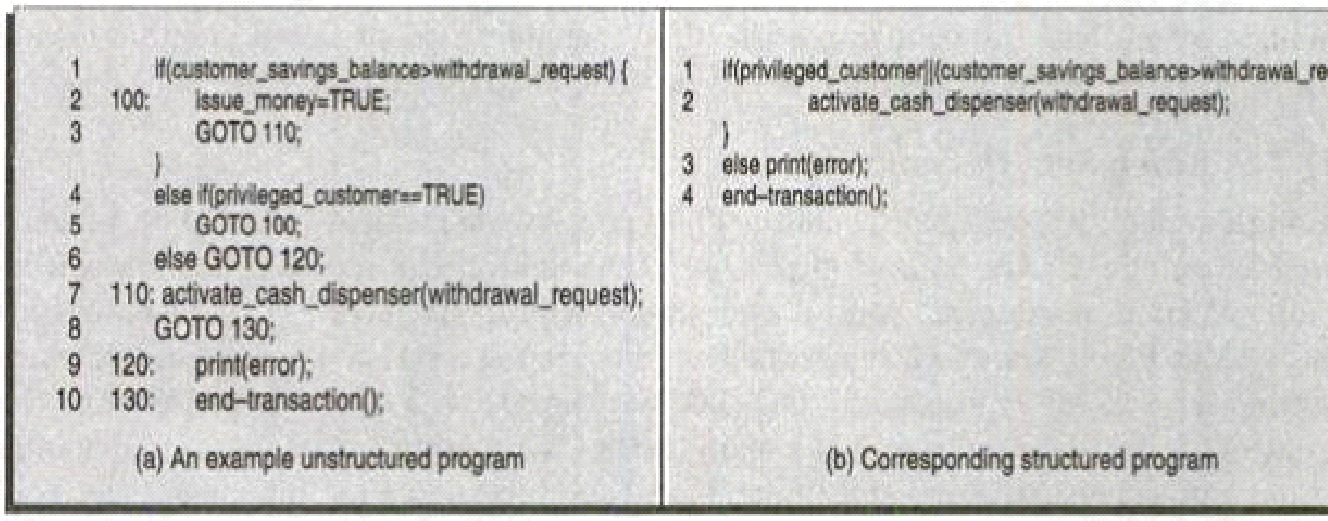
2. Explain emergence of software engineering

EMERGENCE OF SOFTWARE ENGINEERING

Software engineering techniques have evolved over many years in the past. This evolution is the result of a series innovations and experiences.

- **Early Computer Programming**
    - Computers were very slow as compared to today's standards.
    - Programs were very small at that time and lacked sophistication.
    - Programs were written in assembly languages.
    - Programs lengths were typically limited to about few hundred lines of code.
    - Build &fix method is implemented.
    - No proper plan, design or strategy.
    - They went on fixing the problem until they had product worked reasonably well.

- **High Level Language Programming**
    - Computers become faster with the invention of semiconductor technology.
    - High level languages like FORTRAN, ALGOL etc. introduced.
    - Considerably reduce the effort in developing software.
    - Still using build& fix method.
- **Control Flow Based Design**
    - Programs control flow structure indicates the sequence in which the programs instructions are executed.
    - Flow charting technique is developed.
    - If the flowchart representation is simple, then the corresponding code should be simple.



```
1        if(customer_savings_balance>withdrawal_request) {
2   100:      issue_money=TRUE;
3             GOTO 110;
         }
4        else if(privileged_customer==TRUE)
5             GOTO 100;
6        else GOTO 120;
7   110: activate_cash_dispenser(withdrawal_request);
8        GOTO 130;
9   120:      print(error);
10  130:      end-transaction();
```

(a) An example unstructured program

```
1   if(privileged_customer||(customer_savings_balance>withdrawal_re
2            activate_cash_dispenser(withdrawal_request);
    }
3   else print(error);
4   end-transaction();
```

(b) Corresponding structured program

Figure 1.7: Two programs.

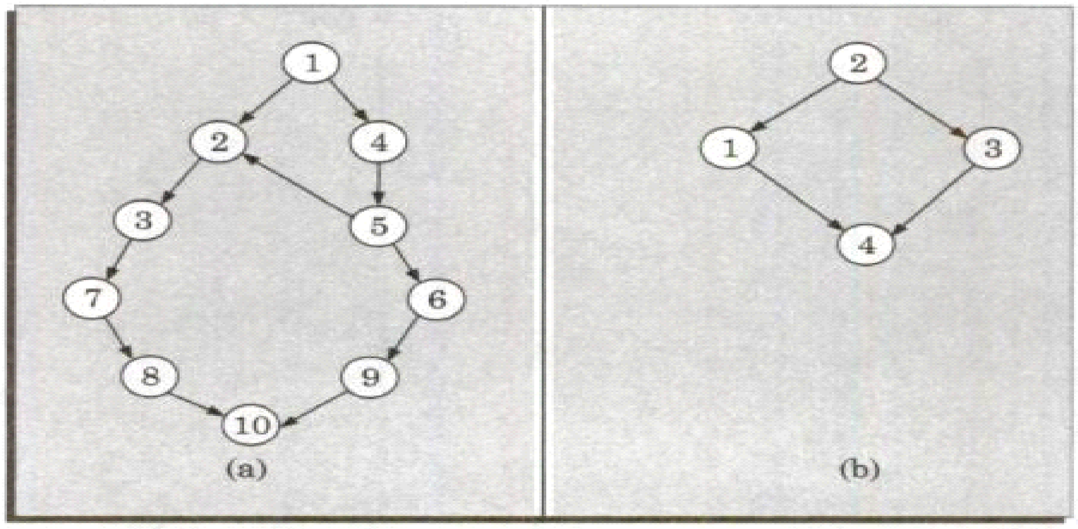


Figure 1.8: Control flow graphs of the programs in Figures 1.7(a) and (b).

- A program having good control flow structure would be easier to understand and develop.
- One understands the program by mentally tracing its execution sequence.
- We can start from a statement producing an output and trace back the statements in the program and understand how they produce the output by transforming the input data.
- We may start with the input data and check by running through how each statement processes the input data until the output is produced.

- **Data Structure oriented Design**
  - Computers became more powerful with the advent of ICs now they use more complex problems.
  - The control flow based program found to be non satisfactory.
  - First programs data structure is designed.
  - Next program design is derived from the data structure.
  - Eg - JSP (Jackson's Structured programming), Warnier- Orr Methodology

- **Data Flow oriented Design**
  - First Major data items handled by a system must be identified and then the processing required on these data items toproduce the desired outputs should be determined.
  - The functions (Processes) and the data items that are exchanged between the different functions are represented in a diagram called DATAFLOW DIAGRAM(DFD).
  - The program structure can be designed from the DFD representation of the problem.
  - Main advantage of DFD is its simplicity.
  - Eg:

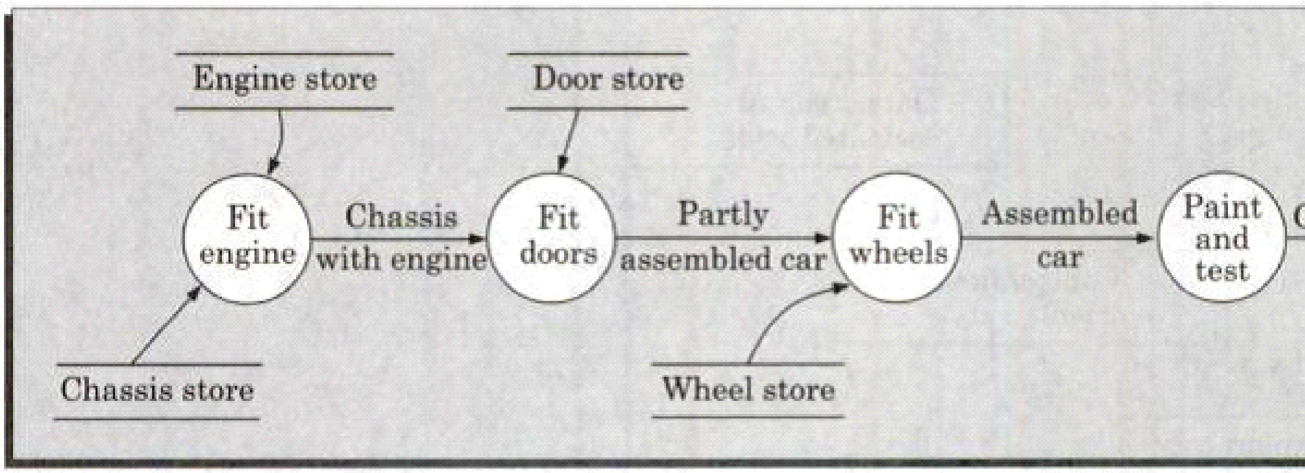


Figure 1.10: Data flow model of a car assembly plant.

- In automated car assembly plant there are several processing stations or work station. Each is specialized to do certain jobs.

- **Object oriented Design**
  - DFD techniques are evolved into Object- oriented design (OOD) techniques.
  - Natural objects (such as employees, pay-roll-register) relevant to the problem are identified.
  - Then relationships among objects such as composition, reference, and inheritance are determined.
  - Each object act as a data hiding(data abstraction) entity
  - Gained acceptance because of simplicity, the scope of code and design reuse.
  - Promises lower development time lower development cost and easier maintenance.
  - The following diagram shows the evolution of software design techniques.

3. Describe Software Process

- A software process also known as a software development life-cycle is composed of a number of clearly defined and distinct work phases which are used to design and build software.

- Development phases represent four fundamental process activities that are common to all software processes:
- *Software specification*. During this activity customers and engineers define the functional and non-functional requirements of software to be produced.
- *Software development.* It is the development activity for designing and implementing software.
- *Software validation*. In this activity the software is checked that it works failure free and to ensure that it provides the functionalities what the customer requires.
- *Software evolution*. The aim of software evolution is to implement changes to the system due to the changes in user and market requirements.

4. State Phases of software development

- Developers should be able to deliver good quality software to the end users using a well defined, well managed, consistent and cost-effective process.
- A software process framework therefore describes the different phases of the project via the activities performed in each phase without telling about the sequence in which these phases or activities will be conducted.
- A software life cycle model is a type of process that represents the order also in which the activities will take place. It describes the sequence of activities graphically to build the final product. As different phases may follow one another, repeat themselves or even run concurrently, the sequence may or may not be linearly sequential.
- Each of these phases will have different activities to meet its objectives.
- The process starts by collecting the user requirements and representing them in a suitable form which must be understandable by developers, analyst, users, testers and all other stake holders of the project. This stage is called the requirements analysis stage and the output of this stage is a document called Requirement Specification Document (RSD) or Software Requirement Specification(SRS).

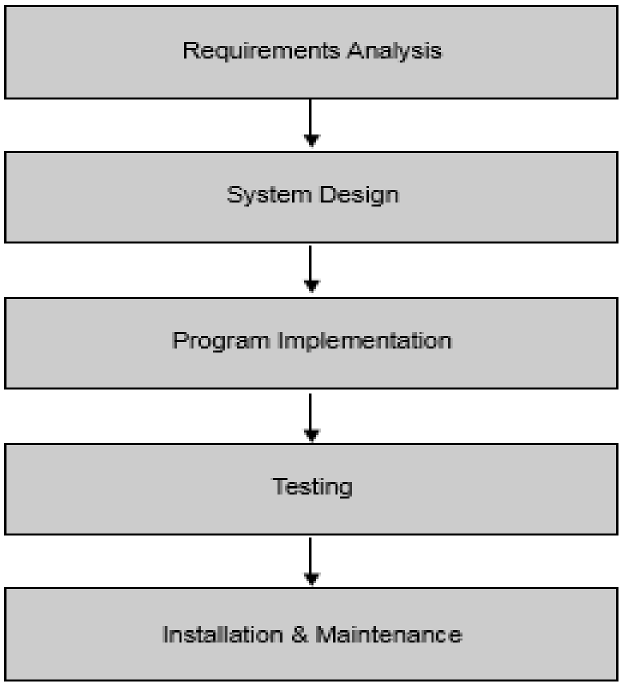


Fig. 1.3 Phases of Software Development Process

- System design stage focuses on high level design of software.
- In program implementation stage actual coding is done and thereafter product is tested to ensure an error free product. Once the product is delivered, it's maintenance is done by the organization.

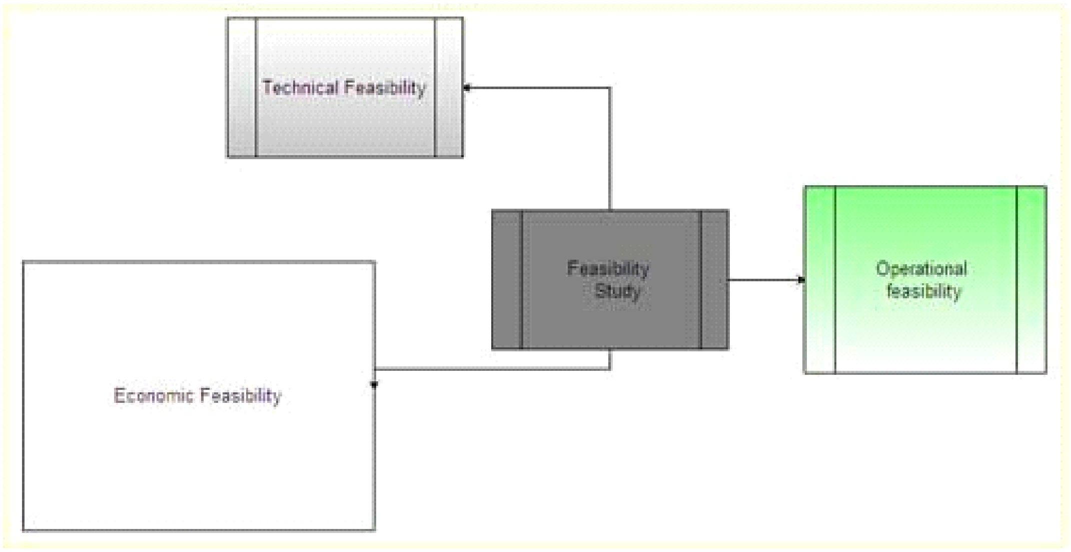
- ***Describe Feasibility study***
  - Feasibility Study is essential to evaluate cost & benefits of the proposed system.
  - On the basis of the fesibility study decision is taken on whether to proceedor to postpone the project or to cancel the project.

  Needs of feasibility study

  - It determined the potential of the existing system.
  - It find or determine all the problem of existing system.
  - To determine all the goals of system.
  - It finds all possible solutions of the problem of existing system.
  - It finds technology required to solve these problems.

- It determines really which solution is easy for operational from the point of view of customer or employees such that it requires very less time with 100% accuracy.
- It determines what hardware and software is required to obtain solution of each problem or proposed system.
- It determines cost requirements of the complete proposed system in terms of cost of hardware required, software required, designing new system, implementation and training, proposed maintenance cost.

- The feasibility study is basically the test of the proposed system in the light of its workability, meeting user's requirements, effective use of resources and of course, the cost effectiveness.
- The main goal of feasibility study is not to solve the problem but to achieve the scope.
- In the process of feasibility study, the cost and benefits are estimated with greater accuracy to find the Return on Investment (ROI).
- Different feasibility study:



- **Economical feasibility -**
- Economic feasibility determines whether the required software is capable of generating financial gains for an organization.
- It involves the cost incurred on the software development team, estimated cost of hardware and software, cost of performing feasibility study, and so on.
- It is essential to consider expenses made on purchases (such as hardware purchase) and activities required to carry out software development.
- In addition, it is necessary to consider the benefits that can be achieved by developing the software.

- Software is said to be economically feasible if it focuses on the issues listed below.
- Cost incurred on software development to produce long-term gains for an organization
- Cost required to conduct full software investigation (such as requirements elicitation and requirements analysis)
- Cost of hardware, software, development team, and training.

There are two types of costs: One time cost and recurring cost.

One time cost involves following:

- Feasibility study cost.
- Cost converting existing system to proposed system.
- Cost to remolding architecture of the office, machinaries, rooms etc.
- Cost of hardware's,
- Cost of operating software's.
- Cost of Application software's.
- Cost of training.
- Cost of documentation preparation.

Recurring cost involve following:

- Cost involves in purchase or rental of equipment.
- Cost of phones and mobile communication equipment.
- Cost of salaries of employee.
- Cost of maintenance of equipment.

- **Operational feasibility –**
- Operational feasibility assesses the extent to which the required software performs a series of steps to solve business problems and user requirements.
- This feasibility is dependent on human resources (software development team) and involves visualizing whether the software will operate after it is developed and be operative once it is installed.
- Operational feasibility also performs the following tasks.
- Determines whether the problems anticipated in user requirements are of high priority
- Determines whether the solution suggested by the software development team is acceptable
- Analyzes whether users will adapt to a new software
- Determines whether the organization is satisfied by the alternative solutions proposed by the software development team.
- **Organizational feasibility –**

- A major computer system makes demands on an organization:
- Does the organization have the management expertise? • Does the organization have the technical expertise?
- Even if the work is carried out by a contractor, the organization needs expertise to oversee the work. • Is the organization committed to the changes in personnel, workflow, etc.?
- Example Copyright deposit system: clerical workflow.

- **Technical feasibility –**
- Technical feasibility assesses the current resources (such as hardware and software) and technology, which are required to accomplish user requirements in the software within the allocated time and budget.
- For this, the software development team ascertains whether the current resources and technology can be upgraded or added in the software to accomplish specified user requirements.
- Technical feasibility also performs the following tasks.
- Analyzes the technical skills and capabilities of the software development team members
- Determines whether the relevant technology is stable and established
- Ascertains that the technology chosen for software development has a large number of users so that they can be consulted when problems arise or improvements are required.

6. Describe Requirement Analysis

- The requirement analyses and identifies what is needed from the system.
- Requirements analysis is done in order to understand the problem the software system is to solve.
- The problem could be automating an existing manual process, developing a new automated system, or a combination of the two.
- The two parties involved in software development are the client and the developer.
- The developer has to develop the system to satisfy the client's needs.
- The developer usually does not understand the client's problem domain, and the client often does not understand the issues involved in software system, This causes a communication gap.
- This phase bridges the gap between the two parties.
- The outcome of this phase is the software requirement specification document (SRS).

- The person responsible for the requirements analysis is often called the analyst.
- Business requirements are gathered in this phase.
- This phase is the main focus of the project managers and stake holders.
- After requirement gathering these requirements in the system to be development is also studied.
- A requirement specification documents is created which serves the purpose of guideline for the next phase of the model.
- The testing team follows the software Testing Cycle and starts the Test Planning phase after the requirements analysis is completed.

7. Describe Design phase

- It is the first step in moving from problem domain to solution domain.
- The purpose of the design phase is to plan a solution of the problem specified by the requirement documents.
- Starting with what is needed, design takes us towards how to satisfy the needs.
- The design of a system is perhaps the most critical factor affecting the quality of the software.
- It has a major impact on the later phases, particularly testing and maintenance.
- The output of this phase is the design document.
- This document is similar to a blueprint or a plan for the solution and is used later during implementation, testing and maintenance.
- It is further of two types:

System Design or Top level Design:	It identifies the various modules that should be in the system, the specifications of the modules and interconnections between the various modules. At the end of system design all the major data structures, file formats, output formats, and the major modules in the system and their specifications are decided.
Detailed Design:	It identifies the internal logic of the various modules. During this

	phase further details of the data structures and algorithmic design of each of the modules is specified. Once the design is complete, most of the major decisions about the system have been made. However, many of the details about coding the designs, which often depend on the programming language chosen, are not specified during design.further details of the data structures and algorithmic design of each of the modules is specified. Once the design is complete, most of the major decisions about the system have been made. However, many of the details about coding the designs, which often depend on the programming language chosen, are not specified during design

- ***Describe Implementation phase***

- After having the user acceptance of the new system developed, the implementation phase begins.
- Implementation is the stage of a project during which theory is turned into practice.
- The major steps involved in this phase are:
- Acquisition and Installation of Hardware and Software
- Conversion
- User Training
- Documentation
- The hardware and the relevant software required for running the system must be made fully operational before implementation.
- The conversion is also one of the most critical and expensive activities in the system development life cycle.
- The data from the old system needs to be converted to operate in the new format of the new system.
- The database needs to be setup with security and recovery procedures fully defined.
- After the users are trained about the computerized system, working has to shift from manual to computerized working. The process is called **Changeover**.
- The following strategies are followed for changeover of the system.

1	**Direct Changeover:**	• This is the complete replacement of the old system by the new system. It is a risky approach and requires comprehensive system testing and training.
2	**Parallel run** :	• In parallel run both the systems, i.e.,

		computerized and manual, are executed simultaneously for certain defined period. The same data is processed by both the systems. This strategy is less risky but more expensive because of the following facts: • Manual results can be compared with the results of the computerized system. • The operational work is doubled. • Failure of the computerized system at the early stage does not affect the working of the organization, because the manual system continues to work, as it used to do.
3	**Pilot run**:	• In this type of run, the new system is run with the data from one or more of the previous periods for the whole or part of the system. The results are compared with the old system results. It is less expensive and risky than parallel run approach. This strategy builds the confidence and the errors are traced easily without affecting the operations.

The documentation of the system is also one of the most important activities in the system development life cycle. This ensures the continuity of the system. Generally following two types of documentations are prepared for any system.

1	**User Documentation**:	The user documentation is a complete description of the system from the user's point of view detailing how to use or operate the system. It also includes the major error messages likely to be encountered by the user.
2	**System Documentation**:	The system documentation contains the details of system design, programs, their coding, system flow, data dictionary, process description, etc. This helps to understand the system and permit changes to be made in the existing system to satisfy new user needs.

9. Describe testing phase

- After the code is developed it is tested against the requirements to make sure that the product is actually solving the needs addressed and gathered during the requirement phase.
- During this phase all types of functional testing like unit testing, integration testing , system testing , acceptance testing are don as well as non-functional testing are also done.

- It is a major quality control measure used during software development.
- Its basic function is to detect errors in the software. Its basic function is to detect errors in the software.
- After the coding phase, computer programs are available that can be executed for testing purpose.
- This implies that testing not only has to uncover errors introduced during coding, but also errors introduced during the previous phases.
- Thus, the goal of testing is to uncover requirement, design, and coding errors in the programs. Consequently, different levels of testing are used.
- The different types of testing are:

a. Unit testing:

It tests each module separately and detects ceding errors.

b. Integration testing:

When the various modules are integrated, this testing is performed to detect errors from the overall system.

c. System testing:

The system is tested against the system requirements to see if all the requirements are met and if the system performs as specified in SRS.

d. User Acceptance Testing:

The system is tested against the user requirements to see if the user is satisfied.

10. Describe Maintenance phase

- Maintenance is necessary to eliminate errors in the system during its working life and to tune the system to any variations in its working environments.
- It must meet the scope of any future enhancement, future functionality and any other added functional features to cope up with the latest future needs.
- It has been seen that there are always some errors found in the systems that must be noted and corrected. It also means the review of the system from time to time.
- The review of the system is done for:
- •knowing the full capabilities of the system

- •knowing the required changes or the additional requirements
- •studying the performance.
- If a major change to a system is needed, a new project may have to be set up to carry out the change.
- The new project will then proceed through all the above life cycle phases.

11. Describe Life Cycle Models- Classical waterfall, Iterative, prototyping, Spiral and Agile

SOFTWARE LIFE CYCLE MODELS

- The life cycle defines a methodology for improving the quality of software and the overall development process.
- Every software product starts with a request for the product by the customer called product conception.
- It is a series of identifiable stages that a software product undergoes during its life time. Or enhance specific software.
- A software life cycle model is a descriptive and diagrammatic representation of the software life cycle.

Advantages:

- Primary advantage of using models is that it helps to develop software in a systematic and planned manner.
- Documentation of the models is necessary in the organization.
- Forms a common understanding of the activities among software engineers.

Phase entry and exit criteria

- Entry and exit of each stage should be clearly mentioned in the development process.
- A phase can start only when the corresponding phase entry criteria are satisfied.
- A phase can be considered to be complete only when exit criteria are satisfied.
- When these two criteria's are satisfied, the next phase can start.

- **Classical Water Fall Model**

- Not a practical model.
- Not used in actual software development.
- Considered as a theoretical way of developing software.
- All other models are based on this model.
- Normally we are using this model for developing software documentation.

Phases of Classical Waterfall Model

- Breaks down the life cycle into an intuitive set of phases.
- The different phases of this model are :
- Feasibility study
- Requirement Analysis & specification
- Design
- Coding & Unit testing
- Integration and System testing
- Maintenance

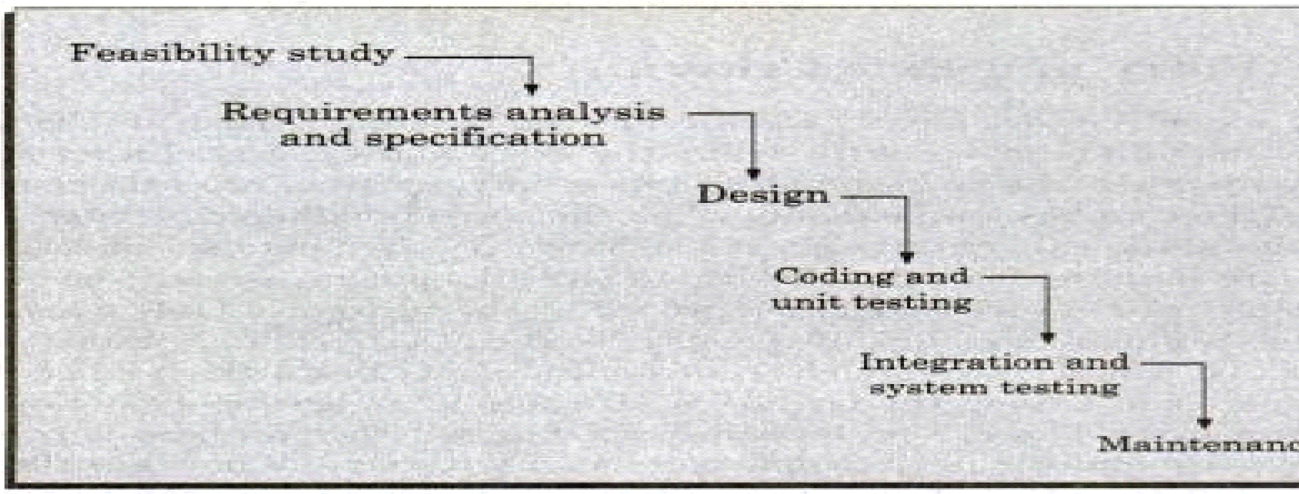
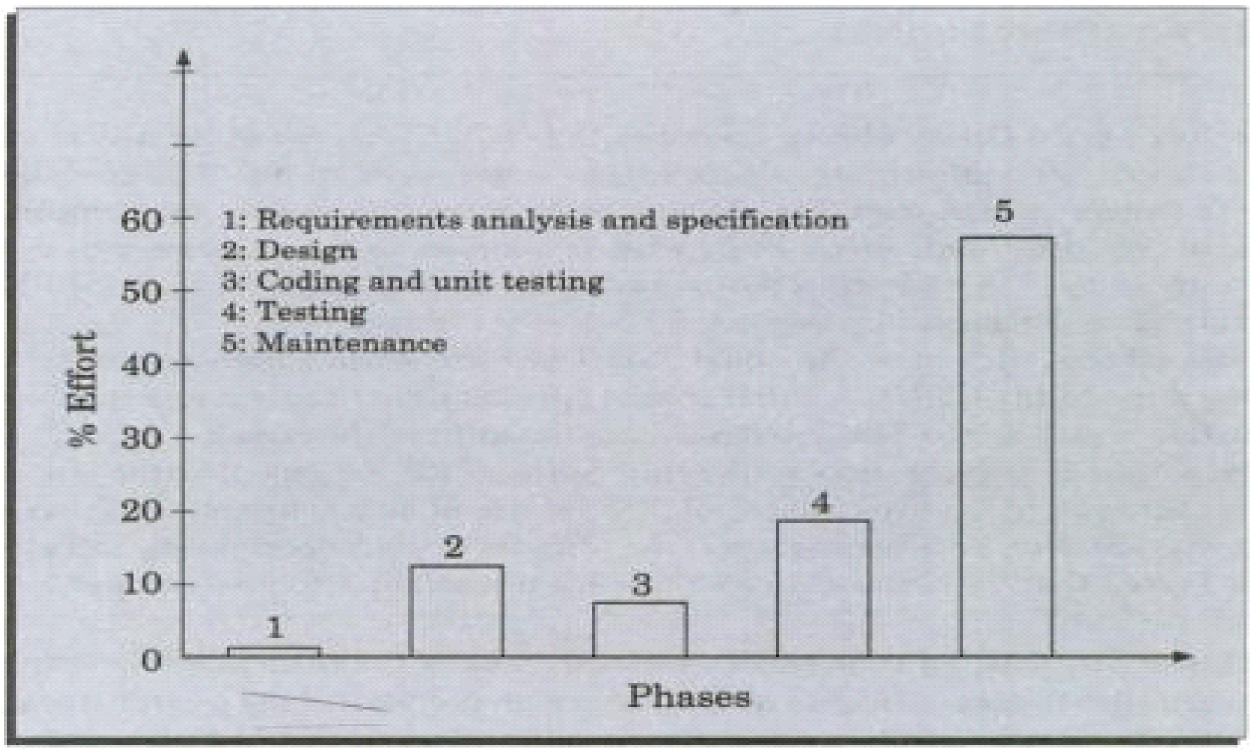

Feasibility study
Requirements analysis and specification
Design
Coding and unit testing
Integration and system testing
Maintenanc

Figure 2.1: Classical waterfall model.

- The work needed to be carried out during different cycle phases typically requires different efforts.



'igure 2.2: Relative effort distribution among different phases of a typical pro

- The phases from feasibility study to integration and system testing are known as *development phases.*
- Software is developed in the development phase and is ready to deliver the customer at the end of the maintenance phase.

- **Feasibility Study**
  - Main aim to check the financial and technical feasibility.
  - Very important stage.
  - Involves analysis of the problem and collection of relevant data.
  - Collected data are analyzed to arrive at the following:
  - **An abstract problem Definition:** Important requirement of the customer are captured and the details of requirement is ignored.
  - **Formulation of the different strategies for solving the problem :**Identifying the different ways for problem solving.

- **Evaluation of the different solution strategies :**Making approximate estimates of the resources, cost of development, time etc and selecting the best solution.

- **Requirement analysis and specification**

- Main aim to understand the exact requirements of the customer and to document them properly.
- Consists of mainly 2 activities :
  - Requirement gathering and analysis
  - Requirement specification

Requirement gathering and analysis

- Gathering the requirement and then analyzing the gathered requirements.
- Goal is to collect all relevant information regarding the product.
- Should clearly understand the customer needs.
- Once requirements are gathered, analysis activity is taken up.

Requirement specification

- Organized in a SRS (Software requirements specification) Document
- Three important content of this document- Functional requirement(includes functions to be supported by system), non functional requirement (includes performance requirements, required standards etc)and goals of implementation.
- SRS document reviewed and approved by the customer.
- Serves as a contract between development team and customer.
- SRS document must be thoroughly understood by developer team and reviewed jointly with customer.

- **Design**

- Goal is to transform requirements specified in SRS document into a structure that is suitable for implementation in programming language.

- Software architecture is derived from SRS.
- Mainly 2 approaches :
    - **Traditional designApproach** :
    - Based on data flow design approach.
    - First structured analysis of the requirement specification is done.
    - And is carried out by structured design activity.
    - DFD’s are used to perform the structured analysis.
    - Structured design activity is undertaken when Structured analysis is completed
    - Structured design consist of 2 main activities
    - Architectural design (high level design): decomposing the system into modules, representing the interfaces.
    - Detailed design (low level design): internals of the modules are designed.
    - **Object Oriented designApproach :**
    - New technique.
    - Objects are identified and then relationships among them are identified.
    - The object structure is further refined to get detailed structure.
    - It has lower development time and effort and maintainability.
- **Coding & Unit testing**
  - Purpose is to translate the software design into source code.
  - Also called as implementation phase since the design is implemented to workable solution.
  - Each component of design is implemented as program module.
  - End product will be set of programs individually tested.
  - After coding is complete each module is unit tested.
  - Unit testing include testing each module in isolation with other modules, then debugging & documenting.

- **<u>Integration & System Testing</u>**
  - Integration of different modules in planned manner.
  - Each integration step, previously planned modules are added to the partially integrated system and the resultant system is tested.
  - After all modules are integrated, system testing is carried out.
  - 3 different kinds of testing:
  - α-testing :system testing performed by development team.
  - β- testing :system testing performed by friendly set of customers.
  - Acceptance Testing : By customer himself after the product delivery to determine whether to accept the delivered product or reject.
  - System testing is done in a planned manner according to *system test plan.*
  - Results are documented as *Test report.*
- **<u>Maintenance</u>**
  - Requires more effort than the effort necessary to develop product.
  - The proportion of maintenance is higher for long lasting projects like OS.
  - Involves 3 activities:
  - **Corrective maintenance** : Correcting errors that are not discovered during the product development phase.
  - **Perfective maintenance:** improving implementations of systems and enhancing the functionalities of the system.
  - **Adaptive maintenance:** Porting the software to the new environment.

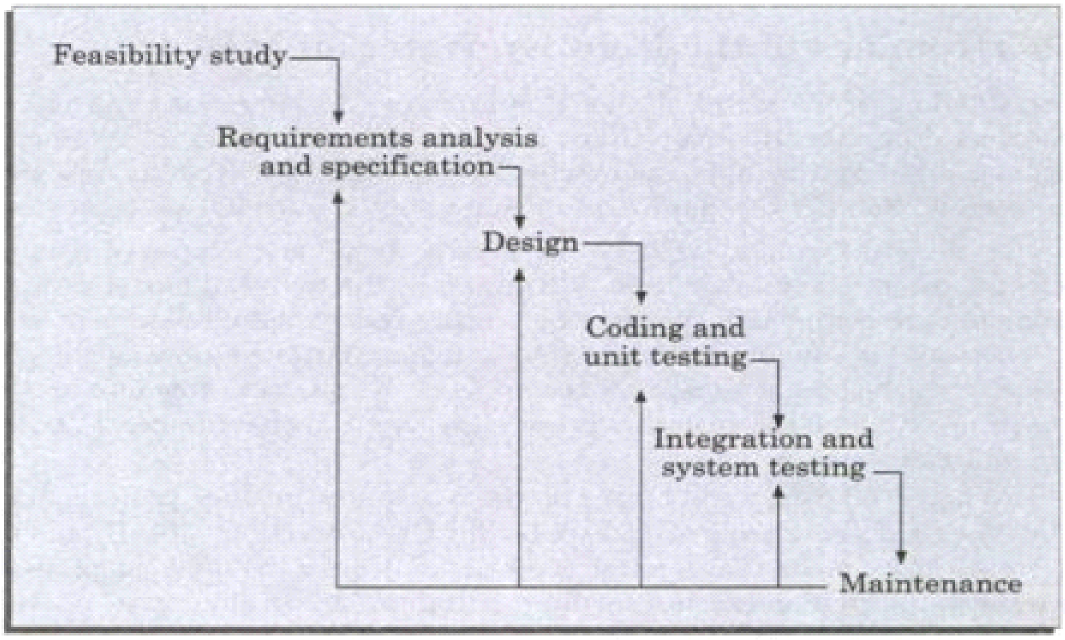
Advantages	**Disadvantages**
Easy to explain to the users	Assumes that the requirements of a system can be frozen.
Structures approach	Very difficult to go back to any stage after it finished.
Stages and activities are well defined	A little flexibility and adjusting scope is

	difficult and expensive.
Helps to plan and schedule the project.	Costly and required more time, in addition to the detailed plan.
Verification at each stage ensures early detection of errors/ misunderstanding.	
Each phase has specific deliverables.	

Shortcomings of Water Fall Model

- No scope of re work at later time because all the activities in phase are flawlessly done.
- Model assumes that all requirements are defined correctly at the beginning of project. On that basis development starts and later requirements were become difficult.
- Assumes all phases are sequential there is a chance to overlap.

- **<u>ITERATIVE WATER FALL MODEL</u>**
- Proposed by making suitable changes in classic water fall model.


Feasibility study
Requirements analysis and specification
Design
Coding and unit testing
Integration and system testing
Maintenance

- The main change is that, this provides feedback paths from every phase to its preceding phases.
- Feedback paths allow correction of errors committed during each phases.
- But feasibility study errors cannot be corrected in later stages of the phases.
- It is developed to overcome the weaknesses of the waterfall model.
- It starts with an initial planning and ends with deployment with the cyclic interactions in between.
- The basic idea behind this model is to develop a system through repeated cycles (iterative) and in smaller portions at a time (incremental), allowing software developers to take advantage of what was learned during development of earlier parts or version of the system.
- **Phase Containment of Errors:** Principle of detecting errors as close to their points of introduction as possible is known as phase containment of errors.

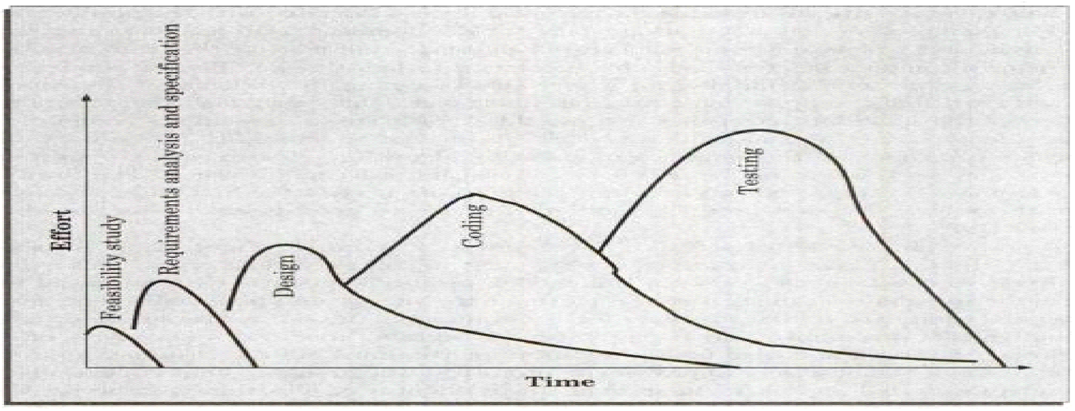


Figure 2.4: Distribution of effort over time for various phases in the iterative waterfall model.

Shortcomings of Iterative Waterfall Model

- Cannot suitably handle the different types of risks that a real life software project may have.
- In real life projects it is difficult to follow the strict phase sequence in this model.

Advantages	Disadvantages

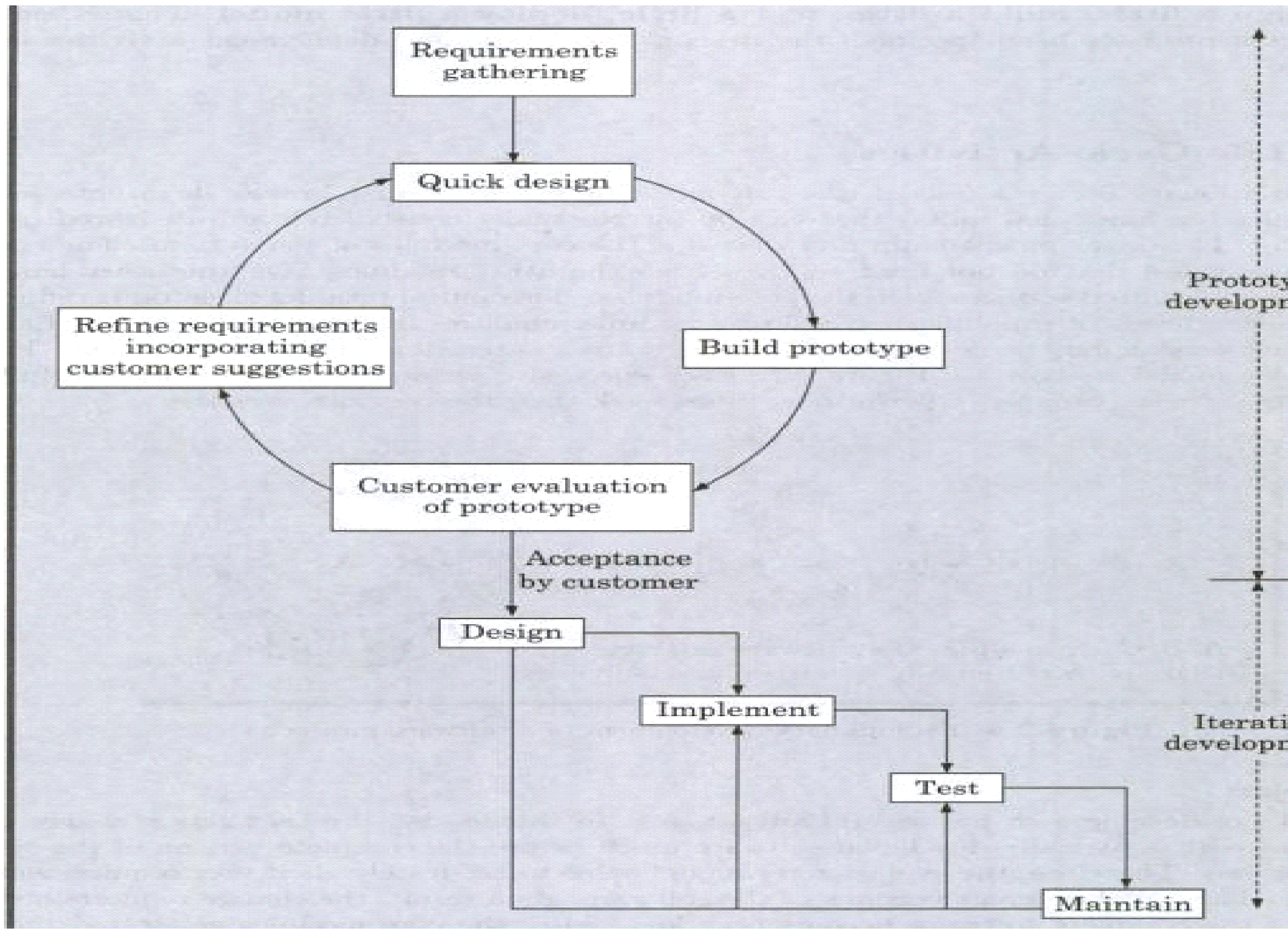
Produce business value early in the development life cycle	Requires heavy documentation Follow a defined set of processes.
More customer involvement	Partitioning the functions and features might be problematic
Better use of scare resources through proper increment definition	Defines increments based on function and feature dependencies.
Can accommodate some change requests between increments	Requires more customer involvement than the linear approaches
More focused on customer value than the linear approaches	Integration between iteration can be an issue if this is not considered during the development.
Problems can be detected earlier	
Lower initial delivery cost and initial product delivery is faster.	

- **PROTOTYPING MODEL**

- Working prototype (toy implementation) of system should be built.
- Built using several dummy functions.
- Prototype usually turns out to be a very simple version, of actual computations with limited functional capabilities.
- Main advantage is that can illustrate the input data format, messages, reports etc to customer
- Mainly used to implement the GUI(Graphical User Interface) part of system
- Can be implemented where the exact technical solution to be adopted are unclear to the team
- It is impossible to get it right the first time; one must plan to develop a good product.
- Starts with requirement gathering phase
- Quick design is made and prototype is built.
- Developed prototype is submitted to customer for feedback.
- Based on the customer feedback modifications, are done.

- Once the customer accepts the prototype, actual system is implemented

Prototyping Model of Software development



Advantages	**Disadvantages**
Reduced time and costs.	Insufficient analysis, guess why?
This can be disadvantage if the developer loses time in developing the prototypes	User confusion of prototypes and finished system.
Improved user involvement	Developer misunderstanding of user objectives
Users give corrective feedback	Excessive development time of the prototype.
A more accurate end products	Expense of implementing prototyping, this can be eliminated if it will be integrated with the final product. Process may continue forever.

- <u>**SPIRAL MODEL**</u>

- It is combing elements of both design and prototyping-in-stages, in an effort to combine advantages of top-down and bottom-up concepts.
- This model of development combines the features of the prototyping model and the waterfall model.
- The spiral model is favored for large, expensive, and complicated projects.
- This model uses many of the same phases as the waterfall model, in essentially the same order, separated by planning, risk Assessment, and the building of prototypes and simulations.
- The diagrammatic representation of this model appears like a spiral with many loops.
- Exact number of loop is not fixed and can vary.
- Each loop is called phase of the software process.
- More flexible compared to other models, since exact number of phases are not fixed.
- Over each loop, one or more features of the product are elaborated and analyzed and the risks at that point of time are identified and are resolved through prototyping. Based on this the identified features are implemented.
- Each phase of this model is split into four sectors or quadrants.

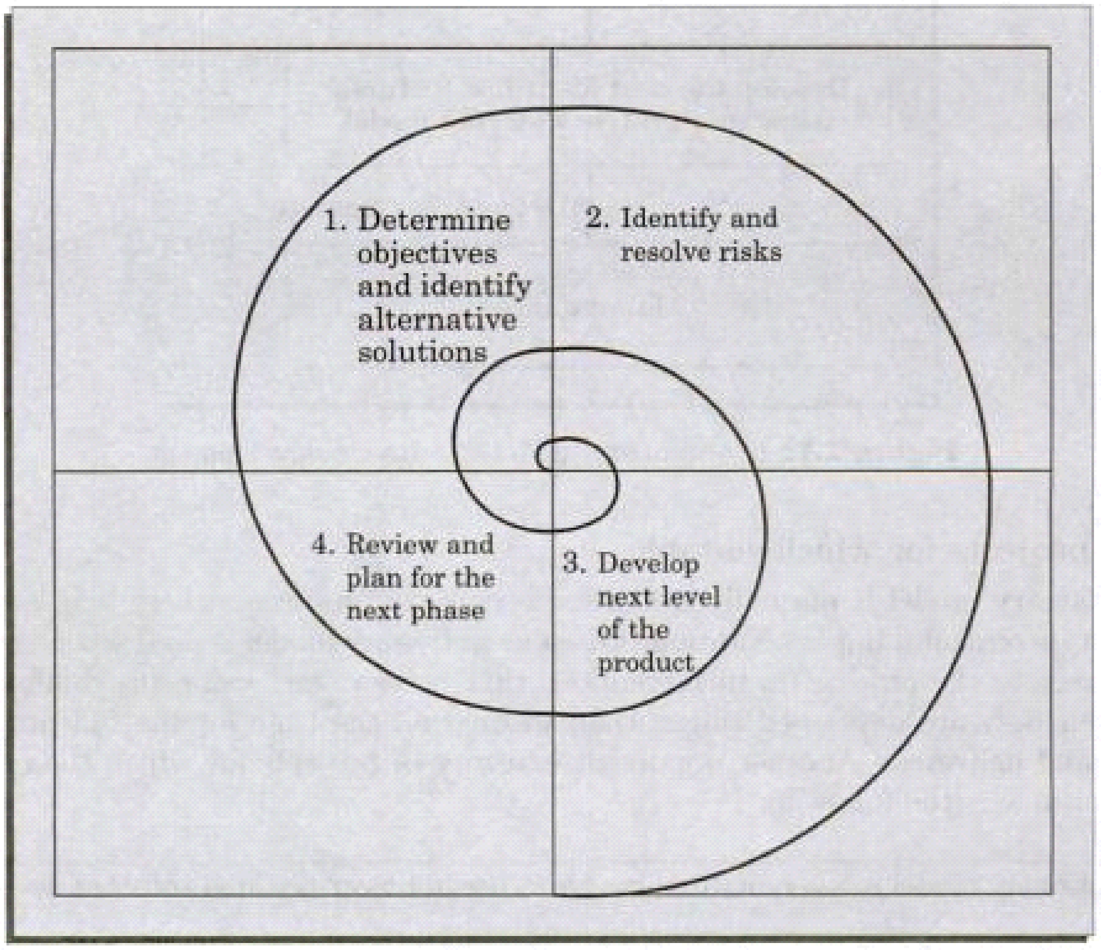

1. Determine objectives and identify alternative solutions
2. Identify and resolve risks
3. Develop next level of the product
4. Review and plan for the next phase

Figure 2.8: Spiral model of software development.

- In first quadrant, some features of the product are identified based on the severity of the risk and how crucial it is to the overall product development.
- Implementation of the identified features forms the objective of the phase
- The objectives are investigated, elaborated, and analyzed.
- Alternative solutions are considered.
- During the second quadrant, solutions are evaluated to select the best possible solution.
- At the end of the third iteration, the identified features have been implemented and the next version of the product is available.

- Results are reviewed in the fourth quadrant
- Suitable for projects having many unknown risks.

Advantages	**Disadvantages**
Estimates become more realistic as work progresses.	High cost and time to reach the final product
Changing requirements can be accommodated	Needs special skills to evaluate the risk and assumptions
Development can be divided into smaller parts and more risky parts can be developed earlier which helps better risk management.	Highly customized limiting re-usability
Users see the system early and give corrective feedback	Time spent planning, setting objectives, doing risk analysis and prototyping will be huge.
A more accurate end product.	Complex model, Requires knowledgeable staffs, Not suitable for outsourced projects.

- **AGIEL MODEL**
  - It is based on iterative and incremental development, where requirements and solutions evolve through collaboration between cross-functional teams.
  - Usually less formal and reduced scope.
  - Used for time-critical applications.
  - An Agile software life cycle is much different as compared to traditional software development frameworks like Waterfall.
  - In Agile, more emphasis is given to sustained and quick development of product features rather than spending more time during the initial project planning, and analyzing the actual requirements.
  - The Agile team develops the product through a series of iterative cycles known as sprints.
  - Besides development activity, other aspects pertaining to development such as product analysis, designing the product features,

developing the functionality, and “testing” the development for bugs are also carried out during the sprints.

- The incremental cycles should always produce a “shippable” product release that can be readily deployed.

- Agile processes make extensive use of events such as the daily scrum meetings, sprint review meeting, and the sprint retrospective meeting to identify and self-correct the development carried out by the team.

- Feedback is solicited frequently, as and when needed, to collaborate, and speed up the development process through sharing of ideas and self-management.

- The feedback system helps to support the self-correction features of Agile frameworks, and is very important.

- The roles played in the Agile process constitute of the product owner, scrum master, and the development team. The product owner “owns” the project on behalf of the stakeholders and ensures that the entire project is developed successfully keeping in mind the stakeholders vision of the product as it should “appear” in the market. The scrum master ensures that the Agile process is followed at all times, and does his or her best to resolve any difficulties or technical issues arising during the development process. The team members participate actively in the daily sprints and make sure meaningful and useful development of product features is presented at all times.

-

Advantages	Disadvantages
Decrease the time required to avail some system features	Scalability
Face to face communication and continuous inputs from customer representative leaves no space for guesswork.	Skill of the software developers
The end result is the high quality software	Ability of customer to express user needs.

in least possible time duration and satisfied customer	
Produces business value early in the development life cycle	Documentation is one at later stages.
	Reduce the usability of components
	Needs special skills for the team.

- **Main features of an Agile software development life cycle model**

  Agile frameworks and methodologies have some common features.

- **1. Individuals and interactions**

  In the Agile software development life cycle model, self-organisation and motivation takes precedence over delegation of authority and following the “seniority” hierarchy. Team members are encouraged to take an active part in the development and planning activities. They are also “empowered” to take certain decisions on their own. The Agile team has to collaborate and share ideas to develop the product “as a whole” unit i.e. each member should support a common vision.

- **2. Working software**

  Agile concentrates upon delivering sustained “working” product releases through product incremental cycles over documentation and working protocols. The main objective is to develop, and deliver, bug free product feature releases in a continuous and sustained manner until the entire product is developed.

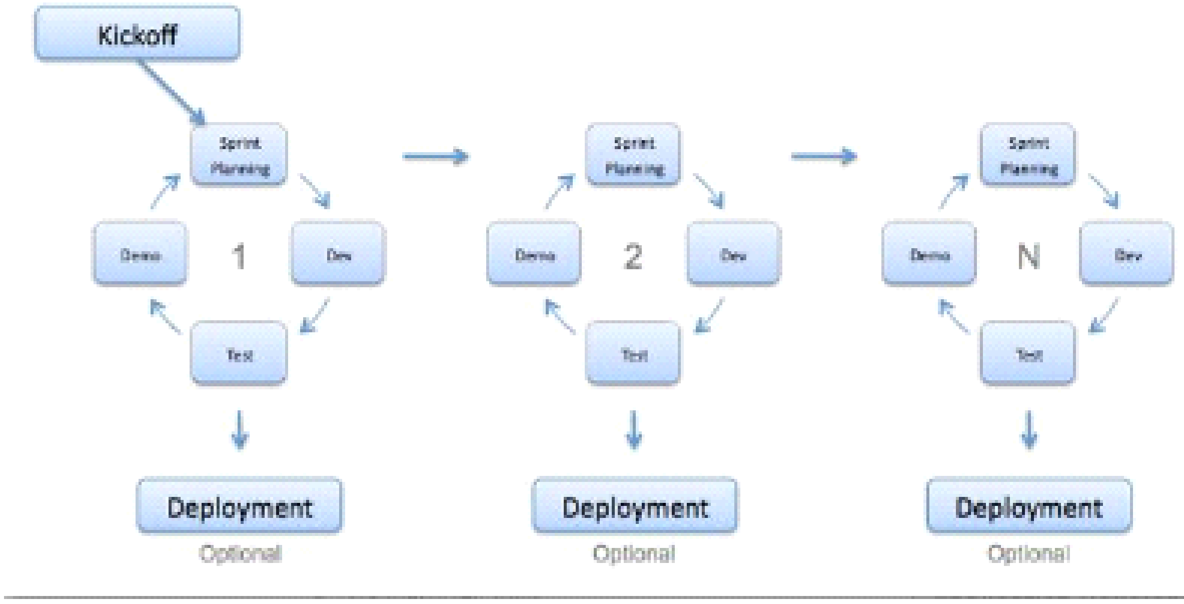
- **3. Customer collaboration**

  Since all the requirements pertaining to product development may not be available, or “acquirable”, at the project start up time owing to various factors, development should commence almost “immediately”, and presented to clients for verification purposes. Stakeholders and project owners “clear” the product features developed through the sprint cycles. A lot of time is saved through customer collaboration, and as a result, the project proceeds in a successful manner as the client always Okays the development keeping in mind the current market trends.

- **4. Responding to changes**

  Agile focuses upon incorporating dynamic changes in the product development cycle. Changes in the product features can be easily and effortlessly carried out by developing “user stories” – product functionality or features as defined in the product backlog. Changes can be carried out at any time while the features are being developed – even late in the product development cycle.

- Explain the agile lifecycle model.
  - The **Agile** software development **life cycle** is based upon the iterative and incremental process models, and focuses upon adaptability to changing product requirements and enhancing customer satisfaction through rapid delivery of working product features and client participation.
  - **Agile development model** is also a type of **Incremental model**. Software is developed in incremental, rapid cycles. This results in small incremental releases with each release building on previous functionality. Each release is thoroughly **tested** to ensure **software quality** is maintained. It is used for time critical applications. Extreme Programming (XP) is currently one of the most well known agile **development life cycle model**.



Advantages of Agile model:

- Customer satisfaction by rapid, continuous delivery of useful software.
- People and interactions are emphasized rather than process and tools. Customers, developers and testers constantly interact with each other.
- Working software is delivered frequently (weeks rather than months).
- Face-to-face conversation is the best form of communication.
- Close, daily cooperation between business people and developers.
- Continuous attention to technical excellence and good design.
- Regular adaptation to changing circumstances.
- Even late changes in requirements are welcomed

Disadvantages of Agile model:

- In case of some software deliverables, especially the large ones, it is difficult to assess the effort required at the beginning of the software development life cycle.
- There is lack of emphasis on necessary designing and documentation.
- The project can easily get taken off track if the customer representative is not clear what final outcome that they want.
- Only senior programmers are capable of taking the kind of decisions required during the development process. Hence it has no place for newbie programmers, unless combined with experienced resources.

COMPARISON OF DIFFERENT LIFE CYCLE MODELS

Factors	**Waterfall**	**Evolutionary**	**Spiral**	**Iterative**	**Agile model**

		Prototyping	**model**	**model**	
Unclear user requirement	poor	Good	excellent	good	Excellent
Unfamiliar technology	Poor	Excellent	Excellent	good	Poor
Complex system	Good	Excellent	Excellent	Good	Poor
Reliable system	good	Poor	Excellent	Good	Good
Short time schedule	poor	Good	Excellent	Excellent	Excellent
Strong project management	Excellent	Excellent	Excellent	Excellent	Excellent
Cost limitation	Poor	Poor	Poor	Excellent	Excellent
Visibility of stakeholders	Good	Excellent	Excellent	Good	Excellent
Skill limitation	Good	Poor	Poor	Good	Poor
documentations	Excellent	Good	Good	Excellent	Poor
Component reusability	excellent	Poor	Poor	Excellent	poor

Classical Water Fall Model

- Basic model
- Not used in practical development

Iterative Classical Water Fall Model

- Most widely used model
- Simple
- Easy to understand
- Suitable for well understood problems
- Not suitable for large projects and projects with more risk

Prototyping Model

- Suitable for projects not well understood
- Used for user interface part of projects

Spiral model

- Meta model and encompasses all other life cycle models.
- Flexibility and risk handling are inherently built into this model
- Suitable for large and technically challenging products.