STUDY MATERIAL

On

SoftwareEngineering

(For 5th Semester CSE)

Prepared by : Mr. Debasish Nanda

Course Contents

7. Software Reliability 115-125

Software Reliability
Different reliability metrics
Reliability growth modeling
Software quality
Software Quality Management System

Model Question for Software Engineering 126-130

BOOKS Recommended:-

Sl.No	Name of Authors	Title of the Book	Name of the publisher
01	Rajib Mall	Fundamentals of Software Engineering	PHI
02	Deepak Jain	Software Engineering: Principles and Practice	Oxford university press
03	Jawadekar	Software Engineering: A Primer	TMH

Chapter – 1

Introduction to Software Engineering

Contents:

- Program vs. Software product
- Emergence of Software Engineering.
- Computer Systems Engineering
- Software Life Cycle Model
- Classical Water fall model
- Iterative Water fall model Prototyping model
- Evolutionary model Spiral model

Relevance of Software Engineering :

- Software engineering is the field of computer science that deals with the building of software systems which are so large or so complex that they arebuild by a team or teams of engineers.
- Parnas has defined software engineering as "multi-person construction of multi-version software".
- According to Fritz Bauer, software engineering is "The establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines".
- Stephen Schach defined as " A discipline whose aim is the production of quality software, software that is delivered on time, within budget, and that satisfies its requirements".
- Software has become critical to advancement in almost all areas of human endeavor. The art of programming only is no longer sufficient to construct large programs.
- There are serious problems in the cost, timeliness, maintenance and quality of many software products.

- The foundation for software engineering lies in the good working knowledge of computer science theory and practice. The theoretical background involves knowing how and when to use data structures, algorithms and understanding what problems can be solved and what cannot. The practical knowledge includes through understanding of the workings of the hardware as well as thorough knowledge of the available programming languages and tools.
- Software engineering has the objective of solving these problems by producing good quality, maintainable software, on time, within budget. To achieve this objective, we have to focus in a disciplined manner on both the quality of the product and on the process used to develop the product.

Software Characteristics and Applications :

Software is a logical rather than a physical system element. Its characteristics that make it different from other things human being build.

- Software is developed or engineered, it is not manufactured in the classical sense which has quality problem.
- Software does not "wear out", but it deteriorates due to change.
- Although the industry is moving toward component-based construction, most software continues to be custom-built. Modern reusable components encapsulate data and processing into software parts to be reused by different programs. E.g. graphical user interface, window, pull-down menus in library etc.

Software Applications :

Software may be applied in any situation for which a pre-specified set of procedural steps has been defined. Information content and determinacy are import factors in determining the nature of a software application. Contents refer to the meaning and form of incoming and outgoing information. Applications are:

- System software: System software is a collection of programs written to service other programs. Examples of system software are compilers, editors, file management utilities, operating system components, drivers.
- Application software: Stand-alone programs for specific needs.
- Engineering / scientific software: Characterized by "number crunching"

algorithms. Such as automotive stress analysis, molecular biology, orbital dynamics etc.

- Embedded software resides within a product or system.
- Product-line software focus on a limited marketplace to address mass consumer market.
- Web based software, the web pages retrieved by a browser are software that incorporates executable instructions and data. As web 2.0 emerges, more sophisticated computing environments is supported integrated with remote database and business applications.
- AI software uses non-numerical algorithm to solve complex problem. Examples are Robotics, expert system, pattern recognition, game playing.

Emergence of Software Engineering:

Software engineering techniques have evolved over many years which resulted series of innovations and accumulation of experience about writing good quality programs. Innovations and programming experiences which have contributed to the development of software engineering are briefly describe in Article 1.4.

Early Computer Programming, High Level Language Programming, Control Flow Based Design, Data Flow Oriented Design, Data Structure Oriented Design, Object and Component Bases Design

Early Computer Programming:

Early commercial computers were very slow as compared to today's standard computers. Even simple processing tasks took more computation time on those computers. No wonder that programs at that time very small in size and lacked sophistication. Those programs were usually written in assembly languages. Program lengths were typically limited to about a few hundreds of lines of monolithic assembly code. Every programmer writing the programs in his own style.

High-Level Language Programming:

Computers become faster with the introduction of the semiconductor technology. With the availability of more powerful computers, it became possible to solve larger and more complex problem. High Level languages such as FORTRAN, ALGOL and COBOL were introduced. This considerably

reduced the effort required to develop software products and helped programmers to write larger programs. However, the software development style was limited to sizes of around a few thousands of lines of source code.

Control Flow-Based Design:

- Programmers found it increasingly difficult not only to write cost effective and correct programs, but also to understand and maintain programs written by others. Thus particular attention is paid to the design of a program's controlflow structure.
- A program's control flow structure indicates the sequence in which the programs instructions are executed.

Data Structure-Oriented Design:

Software engineers were now expected to develop larger more complicated software products which often required writing in excess of several tens of thousands of lines of source code. The control flow-based programs development techniques could not be satisfactorily used to handle these problems and therefore more effective program development techniques were needed. Using data structure-oriented design techniques, first a program's data structures are designed. In the next step, the program design is derived from the data structure.

Object-Oriented Design:

An object-Oriented design technique is an intuitively appealing approach, where the natural objects occurring in a problem are first identified and then the relationships the objects such as composition, reference, and inheritance are determined. Each object essentially acts as a data hiding or data abstraction entry. Object-oriented techniques have gained wide acceptance because of their simplicity, code and design reuse scope they offer and promise of lower development time, lower development cost more robust code and easier maintenance.

Software Life Cycle Models:

- The goal of software engineering is to provide models and processes that lead to the production of well-documented maintainable software.

- A life cycle model prescribes the different activities that need to be carried out to develop a software product and the sequencing of these activities.
- A software life cycle is the series of identifiable stages that a software product undergoes during its lifetime. It also captures the order in which these activities are to be undertaken.
- A software life cycle model is a descriptive and diagrammatic representation of the software life cycle.
- The various phases of software life cycle or Software Development Life Cycle (SDLC) are:
    - ❖ Preliminary Investigation
    - ❖ Software Analysis
    - ❖ Software Design
    - ❖ Software Testing
    - ❖ Software Maintenance
- A software life cycle model is referred to as software process model.

Classical Waterfall Model and Iterative Waterfall Model:

- This model is called as linear sequential model. This model suggests a systematic approach to software development.
- The project development is divided into sequence of well-defined phases. It canbe applied for long-term project and well understood product requirement.
- The classical waterfall model breaks down the life cycle into an intuitive set ofphases. Different phases of this model are:
    - ❖ Feasibility study
    - ❖ Requirements analysis and specification
    - ❖ Design
    - ❖ Coding and unit testing
    - ❖ Integration and system testing
    - ❖ Maintenance

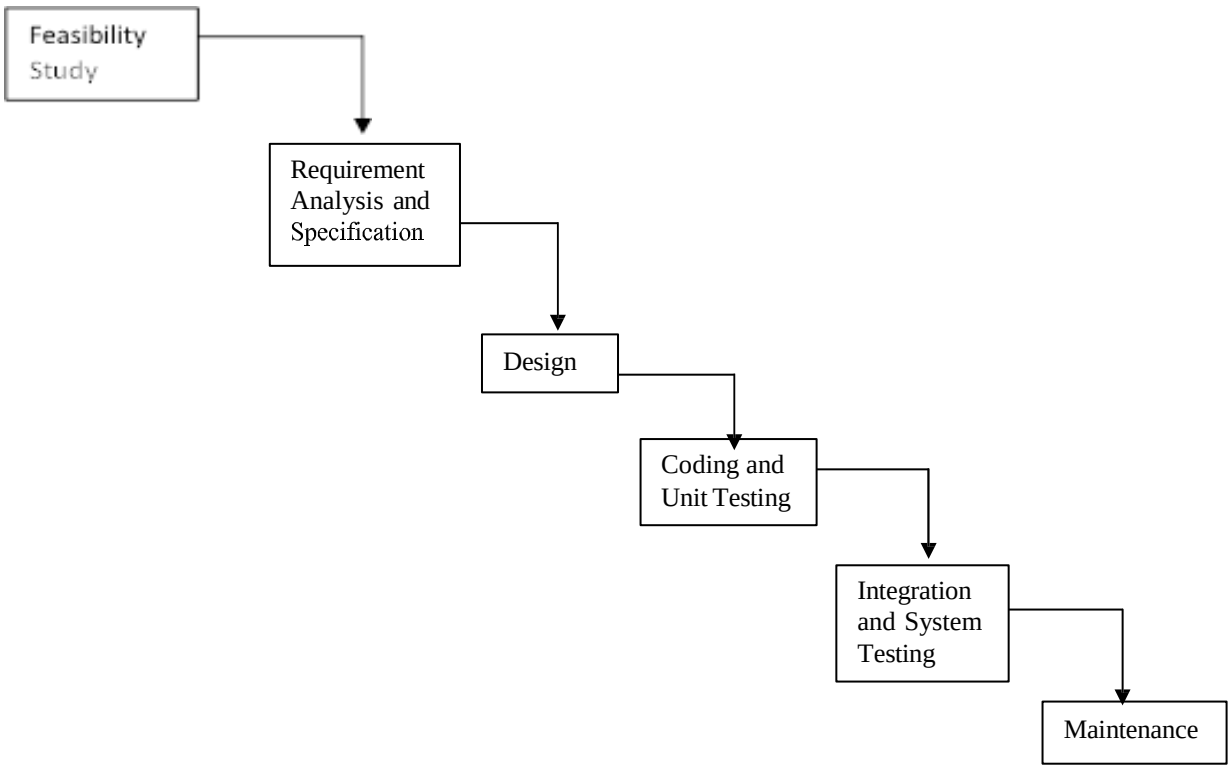

Fig. 1.1 Classical Waterfall Model

The phases starting from the feasibility study to the integration and system testing phases are known as the development phases. All these activities are performed in a set of sequence without skip or repeat. None of the activities can be revised once closed and the results are passed to the next step for use.

Feasibility Study:

The main of the feasibility study is to determine whether it would be financially, technically and operationally feasible to develop the product. The feasibility study activity involves the analysis of the problem and collection of all relevant information relating to the product such as the different data items which would be input to the system, the processing required to be carried out on these data, the output data required to be produced by the system.

Technical Feasibility:

Can the work for the project be done with current equipment, existing software technology and available personnel?

Economic Feasibility:

Are there sufficient benefits in creating the system to make the costs acceptable?

Operational Feasibility:

Will the system be used if it is developed and implemented?

These phases capture the important requirements of the customer, also formulate all the different ways in which the problem can be solved are identified.

Requirement Analysis and Specifications:

The goal of this phase is to understand the exact requirements of the customer regarding the product to be developed and to document them properly.

This phase consists of two distinct activities:

- Requirements gathering and analysis.
- Requirements specification

Requirements Gathering and Analysis:

- This activity consists of first gathering the requirements and then analyzing the gathered requirements.
- The goal of the requirements gathering activity is to collect all relevant information regarding the product to be developed from the customer with a view to clearly understand the customer requirements.
- Once the requirements have been gathered, the analysis activity is taken up.

Requirements Specification:

The customer requirements identified during the requirement gathering and analysis activity are organized into a software requirement specification (SRS) document. The requirements describe the "what" of a system, not the "how". This document written in a natural language contains a description of what the system will do without describing how it will be done. The most important contents of this document are the functional requirements, the non- functional requirements and the goal of implementation. Each function can be

characterized by the input data, the processing required on the input data and the output data to be produced. The non-functional requirements identify the performance requirements, the required standards to be followed etc. The SRS document may act as a contract between the development team and customer.

Design:

The goal of this phase is to transform the requirements specified in the SRS document into a structure that is suitable for implementation in some programming language. Two distinctly different design approaches are being used at present. These are:

- Traditional design approach
- Object-oriented design approach

Traditional Design Approach:

- The traditional design technique is based on the data flow oriented design approach.
- The design phase consists of two activities: first a structured analysis of the requirements specification is carried out, second structured design activity. Structured analysis involves preparing a detailed analysis of the different functions to be supported by the system and identification of the data flow among the functions. Structured design consists of two main activities: architectural design (also called high level design) and detailed design (also called low level design).
- High level design involves decomposing the system into modules, representing the interfaces and the invocation relationships among the modules. Detailed design deals with data structures and algorithm of the modules.

Object-Oriented Design Approach:

- In this technique various objects that occur in the problem domain and the solution domain are identified and the different relationships that exist among these objects are identified.

Coding and Unit Testing:

- The purpose of the coding and unit testing phase of software development is to translate the software design into source code. During testing the

major activities are centered on the examination and modification of the code. Initially small units are tested in isolation from rest of the software product. Unit testing also involves a precise definition of the test cases, testing criteria and management of test cases.

Integration and System Testing:

- During the integration and system testing phase the different modules are integrated in a planned manner. Integration of various modules are normally carried out incrementally over a number of steps. During each integration step previously planned modules are added to the partially integration system and theresultant system is tested. Finally, after all the modules have been successfully integrated and tested system testing is carried out.

The goal of system testing is to ensure that the developed system confirms to its requirements laid out in the SRS document. System testing usually consists of three different kinds of testing activities:

- α –testing: α testing is the system testing performed by the development team.
- β –testing: This is the system testing performed by a friendly set of customers.
- Acceptance testing: This is the system testing performed by the customer himself after the product delivery to determine whether to accept the delivered product or to reject it.

System testing is normally carried out in a planned manner according to a system test plan document. The results of information and system testing are documented in the form of a test report.

Maintenance:

Software maintenance is a very broad activity that includes error correction, enhancement of capabilities and optimization. The purpose of this phase is to preserve the value of the software over time. Maintenance involves performing the following activities:

- Corrective Maintenance

  This type of maintenance involves correcting error that were not discovered during the product development phase.

- Perfective Maintenance

  This type of maintenance involves improving the implementation of the system and enhancing the functionalities of the system according to the customer's requirements.

- Adaptive Maintenance

  Adaptive maintenance is usually required for reporting the softer to work in a new environment.

Iterative Waterfall Model:

- The classical waterfall model is an idealistic one since it assumes that no development error is ever committed by the engineers during any of the life cycle phases. However in practical development environment, the engineers do commit a large number of errors in different phases of the life cycle.

- The sourceof the defects can be wrong assumptions, use of in appropriate technology, communication gap among the project developers etc.

- These defects usually get detected much later in the life cycle. Suppose a defect is detected at testing phase the engineers need to go back to the phase where the defect had occurred and correct the work done during that phase and the subsequent phases to correct the defect and its effect on the later phases.

- In any practical software development work it is not possible to strictly follow the classical waterfall model.

- Feedback paths are needed in the classical waterfall model from every phase to its preceding phases.

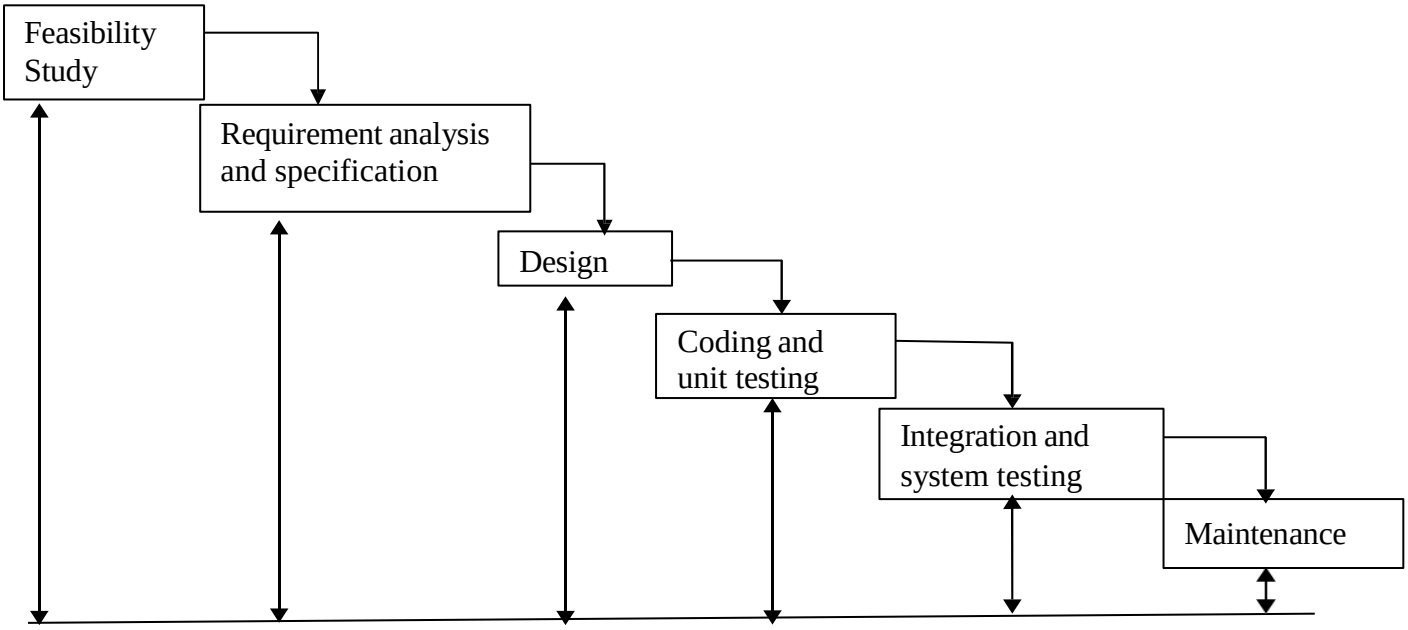

Fig. 1.2 Iterative waterfall Model

- It may not always be possible to detect all error in the same phase in which they occur. The feedback paths allow for correction of the errors committed during a phase, as and when these are detected. If during testing a design error is identified then the feedback path allows the design to reworked and the changes to be reflected in the design documents. However observe that there is no feedback path to the feasibility stage. This means that the feasibility study error can not be corrected.
- Though errors are inevitable in almost every phase of development, it is desirable to detect these errors in the same phase in which they occur. This can reduce the effort required for correcting bugs. The principle of detecting errors as close to there points of introduction as possible is known as phase containment of errors. This is an important software engineering principle.

Prototyping Model:

- Prototyping is an attractive idea for complicated and large systems for which there is no manual process or existing system to help to determine the requirements.
- The main principle of prototyping model is that the project is built quickly to demonstrate the customer who can give more inputs and feedback. This model will be chosen
  - When the customer defines a set of general objectives for

software but does not provide detailed input, processing or output requirements.

- Developer is unsure about the efficiency of an algorithm or the new technology is applied.

- A prototype usually exhibits limited functional capabilities, low reliability and inefficient performance compared to the actual software. A developed prototypecan help engineers to critically examine the technical issues associated with product development.

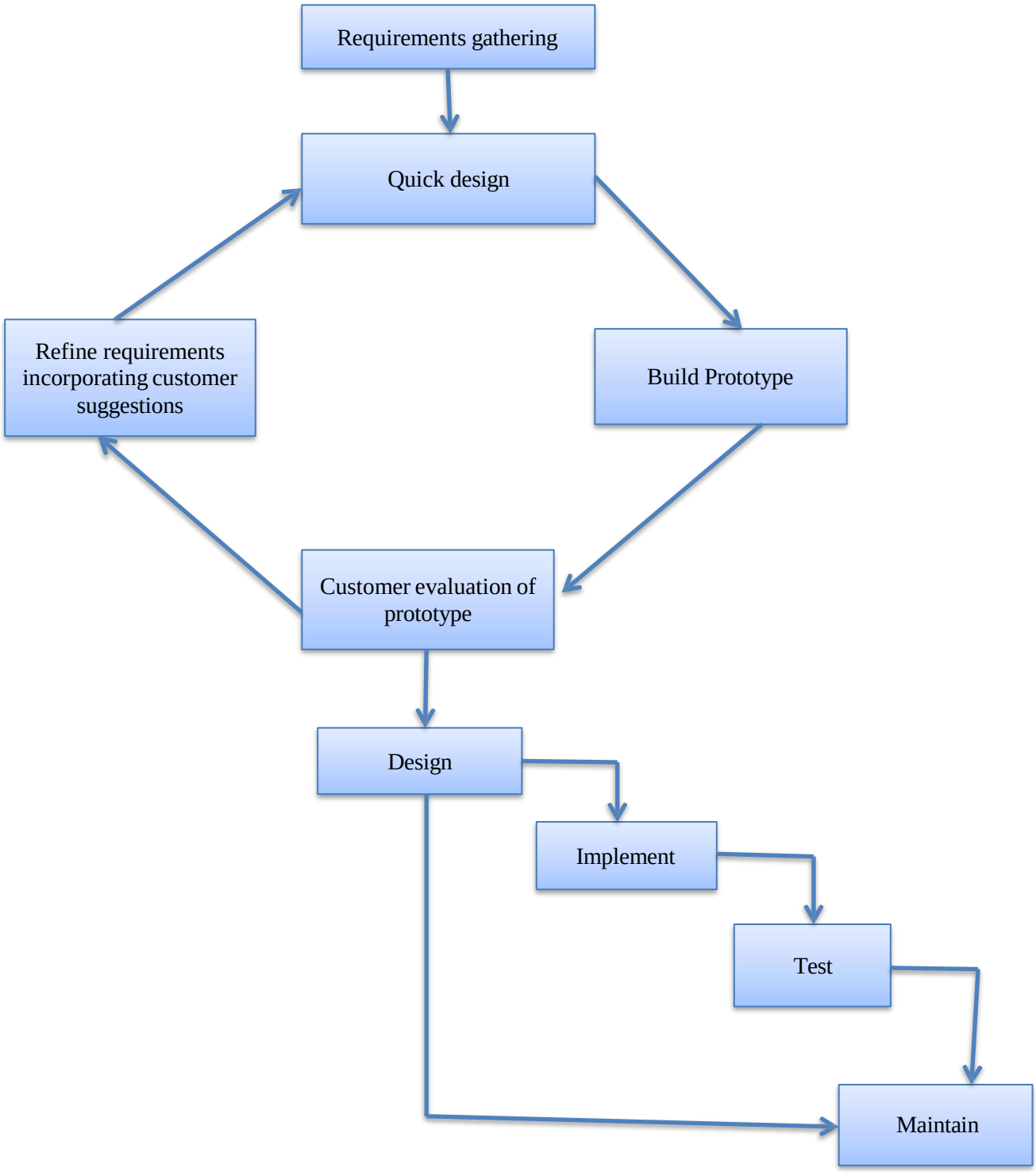

Fig. 1.3 Prototyping Model of Software Development

- The development of the prototype starts when the preliminary version of the requirements specification document has been developed. A quick design is carried out and the prototype is built. The developed prototype is submitted to the customer for his evaluation. Based on the experience, they provide
- feedback to the developers regarding the prototype: what is correct, what needs to be modified, what is missing, what is not needed etc. Based on the customer feedback the prototype is modified and then the users and the clients are again allowed to use the system. This cycle of obtaining customer feedback and modifying the prototype continues till the customer approves the prototype.
- After the finalization of software requirement and specification (SRS) document, the prototype is discarded and actual system is then developed using the iterative waterfall approach.
- Prototyping is often not used, because that development costs may become large. However in some situations, the cost of software development without prototyping may be more than with prototyping. Prototype model is well suited for projects where requirements are hard to determine. This model requires extensive participation and involvement of the customer, which is not always possible.

Evolutionary Model:

This life cycle model is also referred as the successive versions model and the incremental model. In this life cycle model the software is first broken down into several modules or functional units which can be incrementally constructed and delivered.

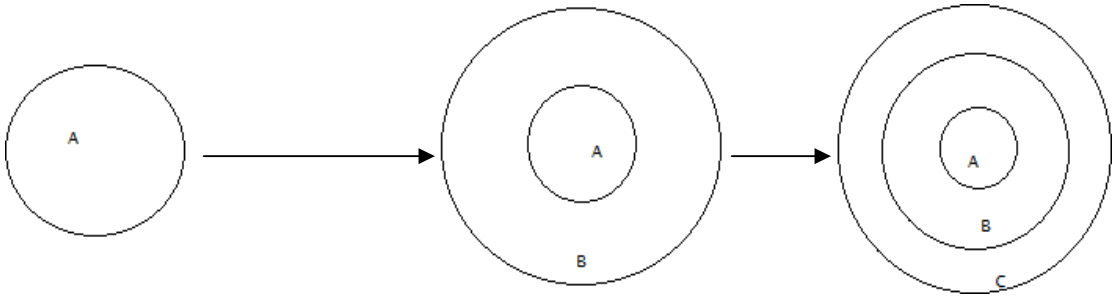

Fig. 1.4 Evolutionary model of software development

- A, B, C are modules of a software product that are incrementally developed anddelivered.
- The development team first develops the core modules of the system. That is basic requirements are addressed but many supplementary features remain undelivered. The initial product is refined into increasing levels of capability by adding new functionalities in successive versions. Each evolutionary version may be developed using an interactive waterfall model of development.

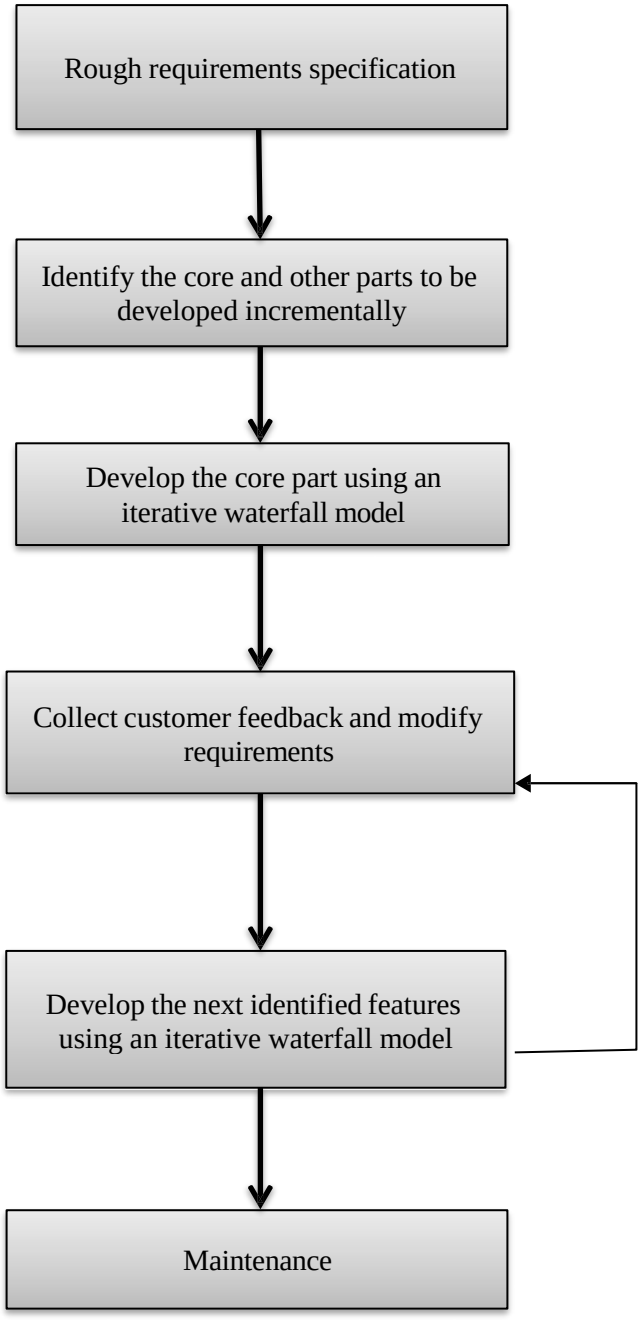

Fig. 1.5 Evolutionary Model of Software Development

- The evolutionary model is used when the customer prefers to receive the products in increments rather than waiting for the full product to be developed and delivered. The evolutionary model is very popular for object oriented software development project.
- The main disadvantage of the successive versions model is that for most practical problems it is difficult to divide the problem into several functional units which can be incrementally implemented and delivered. The evolutionary model is normally useful for only very large products.

Spiral Model:

The spiral model also known as the spiral life cycle model is a systems development life cycle model used in information technology. This model of development combines the features of the prototyping model, the waterfall model and other models. The diagrammatic representation of this model appears like a spiral with many loops.

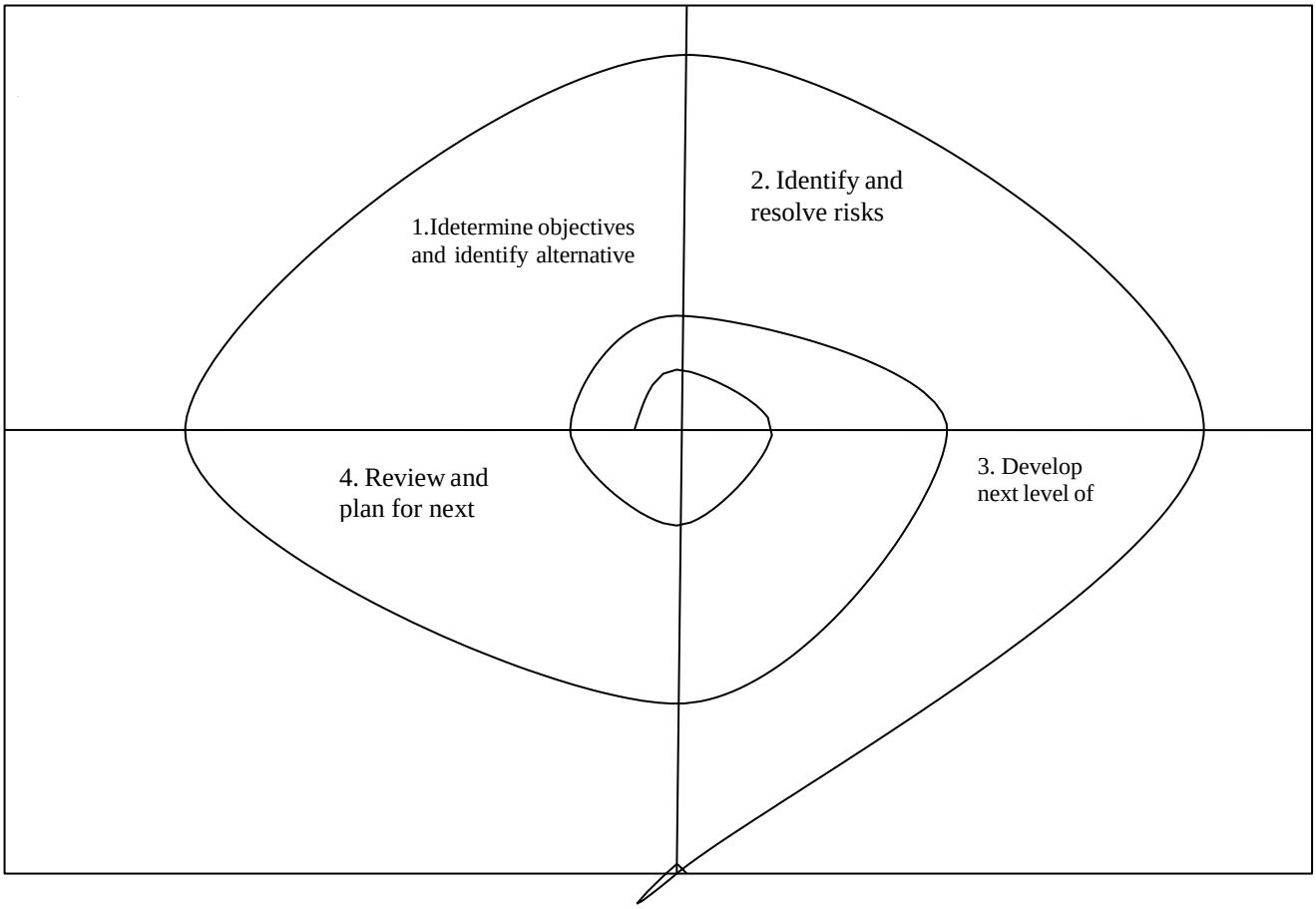

Fig. 1.6 Spiral Model of Software Development

Exact number of phases through which the product is developed in this model is not fixed. The number of phases varies from one project to another. Each phase in this model is split into four sectors or quadrants:

- **Planning:** Identifies the objectives of the phase and the alternative solutions possible for the phase and constraints.
- **Risk analysis:** Analyze alternatives and attempts to identify and resolve the risks involved.
- **Development:** Product development and testing product.
- **Assessment:** Customer evaluation.

During the first phase planning is performed, risks are analyzed, prototypes are built and customers evaluate the prototype. During the second phase a second prototype is evolved by a fourfold procedure: evaluating the first prototype in terms of its strengths, weaknesses and risks, defining the requirements of the second prototype, constructing and testing the second prototype. The existing prototype is evaluated in the same manner as was the previous prototype and if necessary another prototype is developed. After several iterations along the spiral, all risks are resolved and the software is ready for development. At this point, a waterfall model of software development is adopted.

- The radius of the spiral at any point represents the cost incurred in the project till then and the angular dimension represents the progress, made in the current phase.
- In the spiral model of development, the project team must decide how exactly to structure the project into phases. The most distinguishing feature of this model is its ability to handle risks. The spiral model uses prototyping as a risk reduction mechanism and also retains the systematic step-wise approach of the waterfall model.

Spiral Model Strengths:

- Provides early indication of risks, without much cost.
- Critical high-risk functions are developed first.
- Early and frequent feedback from users.
- Cumulative costs assessed frequently.

Spiral Model Weaknesses:

- The model is complex.
- Risk assessment expertise is required.
- May be hard to define objectives.
- Spiral may continue indefinitely.
- Time spent planning, resetting objectives, doing risk analysis and prototyping may be excessive.