UNIT 1 INTRODUCTION TO SOFTWARE PRODUCT, COMPONENT & CHARACTERISTICS

Structure **Page Nos.**

1.0 Introduction 5
1.1 Objectives 5
1.2 Software Product, Components and Characteristics 5
1.3 Software Engineering Phases 7
1.3.1 The Study Phases
1.3.2 The Design Phase
1.3.3 The Development Phase
1.3.4 The Operation Phase
1.4 Documentation of the Software Product 8
1.5 Software Process and Models 9
1.6 Summary 14
1.7 Solutions/Answers 14
1.8 Further Readings 14

1.0 INTRODUCTION

The unit focuses on Software Product, Component and Characteristics. This unit discusses the evolution of Software Engineering life cycle. Waterfall model is discussed in this unit. Software Engineering concepts and their phases are also included in this unit. Software documentation is a continuous and parallel activity in development process.

1.1 OBJECTIVES

After going through this unit, you should be able to know:

- the meaning of a software product;
- the components of a software;
- the characteristics of software;
- about different phases of a software project, and
- about water fall model.

1.2 SOFTWARE PRODUCT, COMPONENTS AND CHARACTERISTICS

We are entering an information age, one in which the management of information resources of organisation will be of vital importance. Business information systems are systems that use these resources to convert data into information in order to improve productivity. Business information systems usually are composed of smaller systems, called subsystems. Computer hardware and software are important resources that support information systems and subsystems.

Systems analysis is a general term that refers to an orderly, structured process for solving problems. Four phases – study, design, development, and operation – make up the life cycle of computer-related business systems. A systems analyst is a person

who performs systems analysis during any or all of the life-cycle phases. The systems analyst not only analyses information system problems, but also synthesises new systems to solve these problems.

The four information eras are: the Early Era (1940-1955), the Growing Era (1965-1995), the Refining Era (1965-1980), and the Maturing Era, 1980 onwards. The Early Era concentrated on hardware, and human-machine communication was very difficult. The growing Era improved this communication through the introduction of English-like programming languages; however, techniques for managing computer-related projects were lacking. During the Refining Era, explosive growth occurred in the development of large (midi and maxi) and small (micro and mini) computer systems and in their applications. Developments in microelectronics technology contributed significantly to this growth. Throughout most of the Refining Era, in spite of a proliferation of applications, difficulties were encountered in using computer to solve business problems. However, towards the end of this era, a structured system analysis process called the life-cycle methodology- came into increasing use as a means of developing usable business information systems. Structured techniques for the analysis, design, and development of computer-related information systems will be enhanced in the maturing Era. These techniques will be used to develop information systems in applications areas such as distributed data processing, the automated office, and management-decision support. This will be an Era in which information will be acknowledged as an important corporate resource. The systems analyst will assume an important role in managing the information resources of the corporation.

The computer-based business system also contains hardware components; however, its most significant characteristics is a software end-product. Software may be defined as a collection of programs or routines that facilitates the use of a computer. This definition includes operational systems, which facilitate the general use computers, and application programs, which are written to solve specific problems. The later is the end-product associated with a computer-based information system. Software, in contrast with hardware, does not possess attributes that can readily be observed and measured from concept to end – product. The software end – product is information. Although it may be stored or printed on a physical medium, such as a magnetic disk, a reel of tape, or a sheet of paper, information is transient and fragile compared with hardware.

Many of the past difficulties in developing effective computer-based business systems stemmed not only from belated efforts to apply management controls, but also from failure to recognise that techniques applicable to the development of hardware end – products could not be applied without modification to the development of software end-products. However, as a result of experience gained from large government and commercial software projects in the latter part of the 1960s and throughout 1970s, the concept of life-cycle management was adapted to fit the development of computer based business systems.

The key to modifying the life-cycle concept for the management of software projects was the recognition that, although supporting documentation accompanies a physical product throughout its development, documentation is the software product.

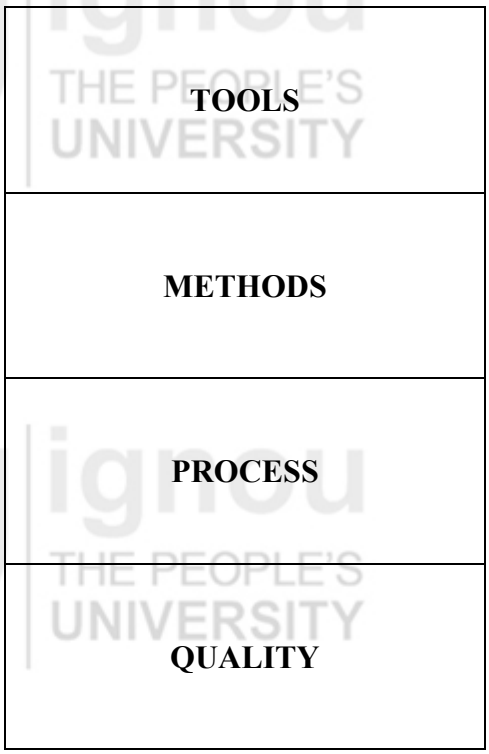

Figure 1.1: Software engineering layers

☞ Check Your Progress 1

1) _____ are systems that use various resources to convert data into information in order to improve productivity.

2) A _____ is a person who performs systems analysis during any or all of the life-cycle phases.

3) _____ may be defined as a collection of programs or routines that facilitates the use of a computer.

1.3 SOFTWARE ENGINEERING PHASES

The life cycle of a computer-based system exhibits the following distinct phases.

The life cycle methodology for developing complex systems is modular and is a top-down procedure. In the study phases, modules that describe the major functions to be performed by the system are developed. The procedure is called top-down because in successive phases the major modules are expanded into additional, increasingly detailed, modules. Powerful graphic tools have been developed to structure the *top-down* design and development phase activities in detail. For present, we can summarise the principal tasks associated with each of the phases of life cycle of a computer-based business system.

1.3.1 The Study Phase

This is the phase during which a problem is identified, alternate system solutions and studied, and recommendations are made about committing the resources required to design the system. Tasks performed in the study phase are grossly analogous to (1) determining that a shelter from the elements is needed, and (2) deciding that a two-

bedroom house is more appropriate shelter than a palace, a cave, or other possible selections.

1.3.2 The Design Phase

In this phase, the detailed design of the system selected in the study phases is accomplished. This is analogous to drawing the plans for the two-bedroom home decided in the study phase. In the case of a computer-based business system, design phase activities include the allocation of resources to different tasks. In the design phase, the technical specifications are prepared for the performance of all allocated tasks.

1.3.3 The Development Phase

This is the phase in which the computer-based system is constructed from the specifications prepared in the design phase. Software is acquired and installed during the development phase. All necessary procedures, manuals, software specifications, and other documentation are completed. The staff are trained, and the complete system is tested for operational readiness. This is analogous to the actual construction of our two-bedroom house from the plans prepared in its design phase. Though the testing phase is different, both were clubbed here only to highlight that most of the software development companies which do coding will also make arrangements for testing. The alternative is that the software development company give the code to another company to test it. This trend is picking up in the software industry.

1.3.4 The Operation Phase

In this phase, the new system is installed and there is a changeover from the old system to the new system. The new system is operated and maintained. Its performance is reviewed, and changes in it are managed. The operation phase is analogous to moving into and loving in the house that we have built. If we have performed the activities of the preceding phases adequately, the roof should not leak.

All of the activities associated with each life-cycle phase must be performed, managed, and documented. Hence, we now define systems analysis as the performance, management, and documentation of the activities related to the four life-cycle phases of a computer based business system. Similarly, we now can identify the systems analyst as the individual who is responsible for the performance of systems analysis of all, or a portion of the phases of the life cycle of a business system.

1.4 DOCUMENTATION OF THE SOFTWARE PRODUCT

The accumulation of documentation parallels the life-cycle performance and management review activities. Documentation is not a task accomplished as a “wind up” activity; rather, it is continuous and cumulative. The most essential documents are called baseline specification (that is, specifications to which change can be referred). There are three baseline specifications:

1. **Performance specification**: This results at the end of study phase. It describes the function of the system to be developed by the user.It is a “design to” specification.

2. **Design specification**: This results at the end of the design phase. It describes the design to the programmers or developers. This throws light on the process of development to them. It is a “build to” specification.

3. **System specification**: This results at the end of the development phase and consists of all the system documentation. It is the basis for all manuals and procedures, and it is an "as built" specification.

The design specification evolves from the performance specification, and the system specification evolves from the design specification. Since these documents are the only measurable evidence that progress is being made towards the creation of a useful software end-product, it is not possible to manage the life-cycle process without them. Thus, documentation is not only the "visible" software end product, but also the key to successful management of the life cycle of computer-based business systems.

1.5 SOFTWARE PROCESS AND MODELS

From the inception of proposal for a software system, until it is implemented and delivered to a customer, and even after that, the system undergoes gradual development and evolution. The software is said to have a life cycle composed of several phases. In the traditional life cycle model, called the "waterfall model" each phase has well-defined starting and ending points, with clearly identifiable deliverables to the next phase.

The following are some of the models adopted to develop software:

(i) Build and Fix Model

It is a simple two phase model. In one phase, code is developed and in another, code is fixed.

Figure 1.2 depicts the Build and Fix model.

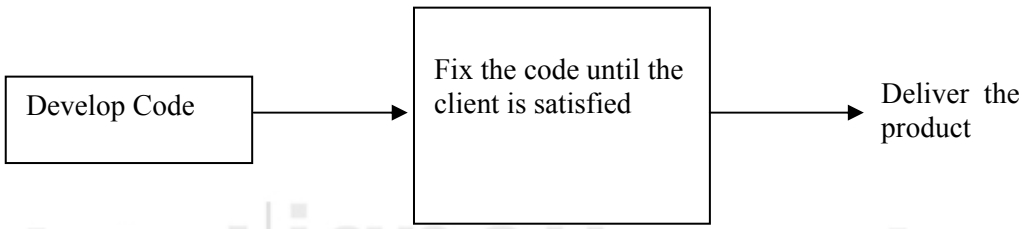

Figure 1.2 : Build and fix model

(ii) Waterfall Model

It is the simplest, oldest and most widely used process model. In this model, each phase of the life cycle is completed before the start of a new phase. It is actually the first engineering approach of software development.

Figure 1.3 depicts Water Fall Model.

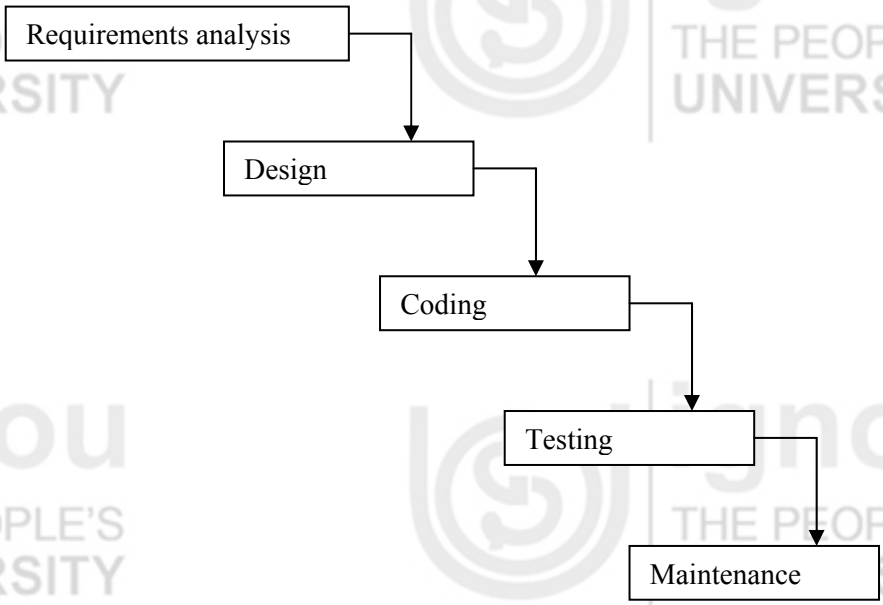

Figure 1.3 : Water fall model

The functions of various phases are discussed in software process technology.

The waterfall model provides a systematic and sequential approach to software development and is better than the build and fix approach. But, in this model, complete requirements should be available at the time of commencement of the project, but in actual practice, the requirements keep on originating during different phases. The water fall model can accommodate the new requirements only in maintenance phase. Moreover, it does not incorporate any kind of risk assessment. In waterfall model, a working model of software is not available. Thus, there is no methods to judge the problems of software in between different phases.

A slight modification of the waterfall model is a model with feedback. Once software is developed and is operational, then the feedback to various phases may be given.

Figure 1.4 depicts the Water Fall Model with feedback.

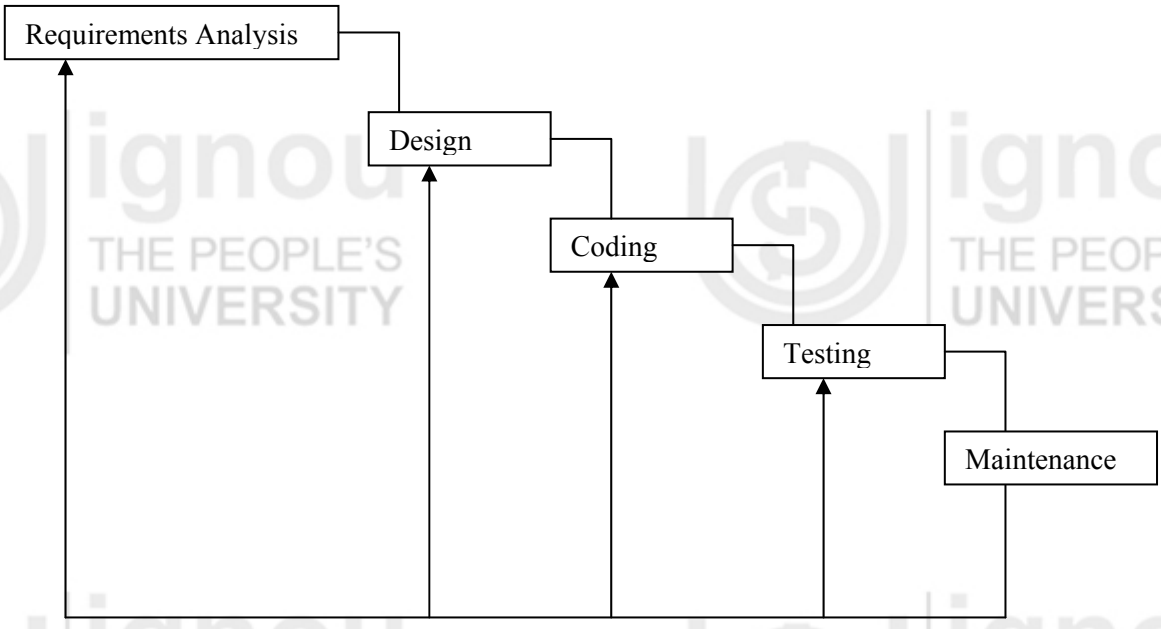

Figure 1.4 : Water fall model with feedback

(iii) Iterative Enhancement Model

This model was developed to remove the shortcomings of waterfall model. In this model, the phases of software development remain the same, but the construction and delivery is done in the iterative mode. In the first iteration, a less capable product is developed and delivered for use. This product satisfies only a subset of the requirements. In the next iteration, a product with incremental features is developed. Every iteration consists of all phases of the waterfall model. The complete product is divided into releases and the developer delivers the product release by release.

Figure 1.5 depicts the Iterative Enhancement Model.

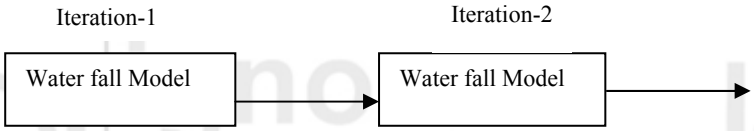

Figure 1.5: Iterative enhancement model

This model is useful when less manpower is available for software development and the release deadlines are tight. It is best suited for in-house product development, where it is ensured that the user has something to start with. The main disadvantage of this model is that iteration may never end, and the user may have to endlessly wait for the final product. The cost estimation is also tedious because it is difficult to relate the software development cost with the number of requirements.

(iv) Prototyping Model

In this model, a working model of actual software is developed initially. The prototype is just like a sample software having lesser functional capabilities and low reliability and it does not undergo through the rigorous testing phase. Developing a working prototype in the first phase overcomes the disadvantage of the waterfall model where the reporting about serious errors is possible only after completion of software development.

The working prototype is given to the customer for operation. The customer, after its use, gives the feedback. Analysing the feedback given by the customer, the developer refines, adds the requirements and prepares the final specification document. Once the prototype becomes operational, the actual product is developed using the normal waterfall model.

The prototype model has the following features:

(i) It helps in determining user requirements more deeply.
(ii) At the time of actual product development, the customer feedback is available.
(iii) It does consider any types of risks at the initial level.

Figure 1.6 depicts the prototyping model.

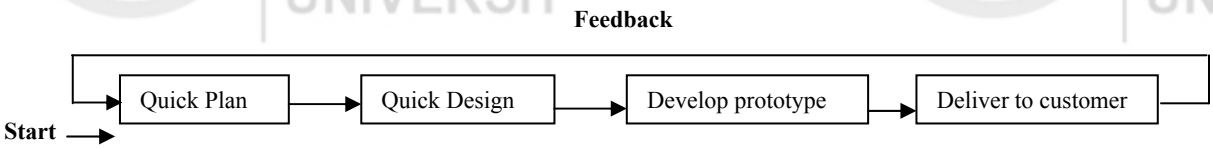

Figure 1.6 Prototyping model

(v) Spiral Model

This model can be considered as the model, which combines the strengths of various other models. Conventional software development processes do not take uncertainties

into account. Important software projects have failed because of unforeseen risks. The other models view the software process as a linear activity whereas this model considers it as a spiral process. This is made by representing the iterative development cycle as an expanding spiral.

The following are the primary activities in this model:

- **Finalising Objective:** The objectives are set for the particular phase of the project.
- **Risk Analysis:** The risks are identified to the extent possible. They are analysed and necessary steps are taken.
- **Development:** Based on the risks that are identified, an SDLC model is selected and is followed.
- **Planning:** At this point, the work done till this time is reviewed. Based on the review, a decision regarding whether to go through the loop of spiral again or not will be decided. If there is need to go, then planning is done accordingly.

In the spiral model, these phases are followed iteratively.
Figure 1.7 depicts the Boehm's Spiral Model (IEEE, 1988).

Figure 1.7: Spiral model

In this model Software development starts with lesser requirements specification, lesser risk analysis, etc. The radical dimension this model represents cumulative cost. The angular dimension represents progress made in completing the cycle.

The inner cycles of the spiral model represent early phases of requirements analysis and after prototyping of software, the requirements are refined.

In the spiral model, after each phase a review is performed regarding all products developed upto that point and plans are devised for the next cycle. This model is a realistic approach to the development of large scale software. It suggests a systematic approach according to classical life cycle, but incorporates it into iterative framework. It gives a direct consideration to technical risks. Thus, for high risk projects, this model is very useful. The risk analysis and validation steps eliminate errors in early phases of development.

(vi) RAD Approach

As the name suggests, this model gives a quick approach for software development and is based on a linear sequential flow of various development processes.

The software is constructed on a component basis. Thus multiple teams are given the task of different component development. It increases the overall speed of software development. It gives a fully functional system within very short time. This approach emphasises the development of reusable program components. It follows a modular approach for development. The problem with this model is that it may not work when technical risks are high.

Software Requirements Specification (SRS)

This document is generated as output of requirement analysis. The requirement analysis involves obtaining a clear and thorough understanding of the product to be developed. Thus, SRS should be consistent, correct, unambiguous & complete, document. The developer of the system can prepare SRS after detailed communication with the customer. An SRS clearly defines the following:

- External Interfaces of the system: They identify the information which is to flow '*from and to'* to the system.
- Functional and non-functional requirements of the system. They stand for the finding of run time requirements.
- Design constraints:

The SRS outline is given below:

1. Introduction
   - 1.1 Purpose
   - 1.2 Scope
   - 1.3 Definitions, acronyms, and abbreviations
   - 1.4 References
   - 1.5 Overview
2. Overall description
   - 2.1 Product perspective
   - 2.2 Product functions
   - 2.3 User characteristics
   - 2.4 Constraints
   - 2.5 Assumptions and dependencies
3. Specific requirements
   - 3.1 External Interfaces
   - 3.2 Functional requirements
   - 3.3 Performance requirements
   - 3.4 Logical Database requirements
   - 3.5 Design Constraints
   - 3.6 Software system attributes
   - 3.7 Organising the specific requirements
   - 3.8 Additional Comments
4. Supporting information
   - 2.1 Table of contents and index
   - 2.2 Appendixes

☞ Check Your Progress 2

1) The purpose of _____ is to identify and document the exact requirements for the system.

2) _____ deals with overall module structure and organisation, rather than the details of the modules.

1.6 SUMMARY

As presented above, the phases give a partial, simplified view of the conventional waterfall life cycle model. The process may be decomposed into a different set of phases, with different names, different purposes, and different granularity. Entirely different life cycle schemes may even be proposed, not based on a strictly phased waterfall development. For example, it is clear that if any tests uncover defects in the system, we have to go back at least to the coding phases and perhaps to the design phase to correct some mistakes. In general, any phase may uncover problems in previous phase. This will necessitate going back to earlier phases and redoing some work. For example, if the system design phase uncovers inconsistencies or ambiguities in the system requirements, the requirements analysis phase must be revisited to determine what requirements were really intended.

Another simplification in the above presentation is that it assumes that a phase is completed before the next one begins. In practice, it is often expedient to start a phase before a previous one is finished.

Most books on software engineering are organised according to the traditional software life cycle model, each, section or chapter being devoted to one phase. Once mastered, the software engineer can apply these principles in all phases of software development, and also in life cycle models that are not based on phased development, as discussed above. Indeed, research and experience over the past decade have shown that there is a variety of life cycle models and that no single one is appropriate for all software systems.

1.7 SOLUTIONS / ANSWERS

Check Your Progress 1

1) Business Information Systems
2) Systems Analyst
3) Software

Check Your Progress 2

1) Requirements Analysis
2) High level design

1.8 FURTHER READINGS

1) *Software Engineering, Sixth Edition,* Ian Sommerville; Pearson Education.
2) *Software Engineering – A Practitioner's Approach*, Roger S. Pressman; McGraw-Hill International Edition.

Reference websites

- **http://www.rspa.com**
- **http://www.ieee.org**
- **http://standards.ieee.org**