SOFTWARE ENGINEERING FOR DESIGN OF MEDICAL DEVICES

7.1 INTRODUCTION

Software: is defined as a collection of programs, procedures, rules, data and associated documentation. The s/w is developed keeping in mind certain h/w and operating system consideration commonly known as platform. And engineering means systematic procedure to develop software.

IEEE definition of Software engineering: A systematic, disciplined and quantifiable approach to the development, operation, maintenance and refinement of software.

Important of software are due to much reason as it is used in:

i) Business decision making

Ex- accounting s/w, billing s/w

ii) For scientific research & engineering problem solving.

Ex-weather forecasting system, space research s/w

iii) It is embedded in multifunctional systems such as medical, telecom entertainment etc.

Ex-s/w for medical patient automation, s/w of GSM/CDMA service provides.

7.2 Software Quality

Several quality factors associated with software quality are as following:

· **Portability:** A software product is said to be portable, if it can be easily made to work in different operating system environments, in different machines, with other software products, etc.

· **Usability:** A software product has good usability, if different categories of users (i.e. both expert and novice users) can easily invoke the functions of the product.

Reusability: A software product has good reusability, if different modules of the product can easily be reused to develop new products.

· **Correctness:** A software product is correct, if different requirements as specified in the SRS document have been correctly implemented.

· **Maintainability:** A software product is maintainable, if errors can be easily corrected, new functions can be easily added to the product, and the functionalities of the product can be easily modified, etc.

7.3 Types of software:-

Computer s/w is mainly divided into two types.

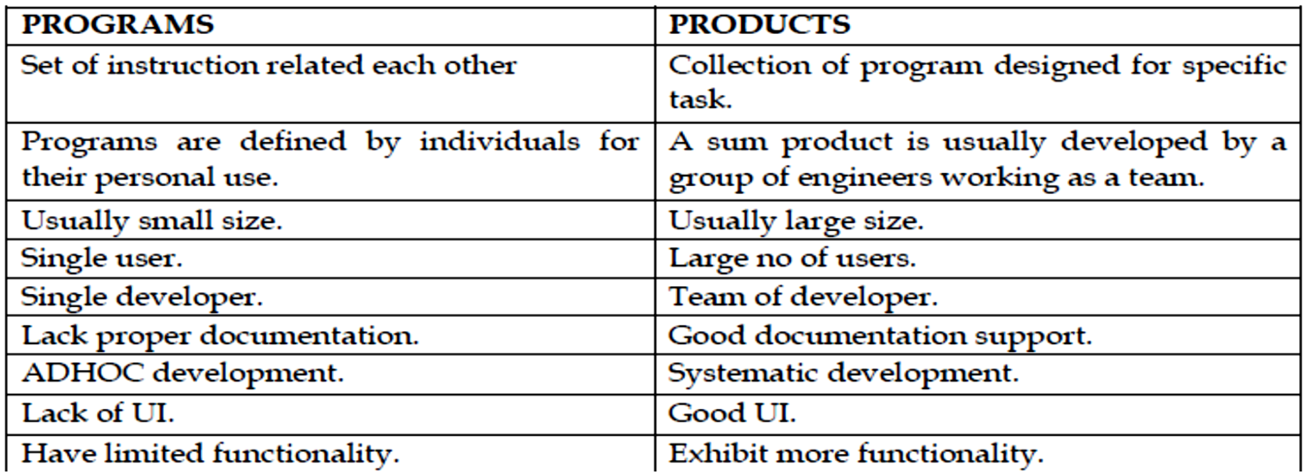
a) system s/w

Application s/w consists of programs to perform user oriented tasks. System s/w includes the operating system & all the utilities to enable the computer to run. Ex: window operating system

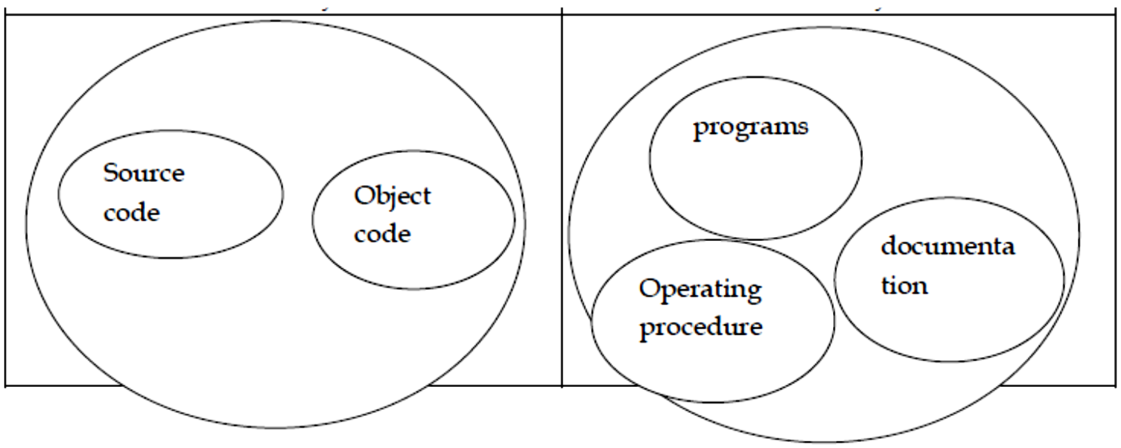
b) application s/w

Application s/w consists of programs to perform user oriented tasks. Ex-word processor, database management. Application s/w sits about the system s/w because it needs help of the system s/w to run.

7.4 Program vs Software product



PROGRAMS	PRODUCTS
Set of instruction related each other	Collection of program designed for specific task.
Programs are defined by individuals for their personal use.	A sum product is usually developed by a group of engineers working as a team.
Usually small size.	Usually large size.
Single user.	Large no of users.
Single developer.	Team of developer.
Lack proper documentation.	Good documentation support.
ADHOC development.	Systematic development.
Lack of UI.	Good UI.
Have limited functionality.	Exhibit more functionality.



7.4.1 Types of software products:

-Generic products: [This type of software product are developed by a organization and sold on open market to any customer], (System software,, application software)

-Customized (or bespoke) products: This type of software products are developed by a software contractor and especially for a customer.

-Embedded Product: Combination of both hardware and software

7.4.2 Software Engineering

Application of engineering for development of software is known as software engineering. It is the systematic, innovative technique and cost effective approach to develop software. And person involved in developing product is called software engineer. S/w engineer is a licensed professional engineer who is skilled in engineering discipline.

7.4.3 Qualities / Skills possessed by a good software engineer:

1. **General Skill** (Analytical skill, Problem solving skill, Group work skill)
2. **Programming Skill** (Programming language , Data structure , Algorithm , Tools(Compiler, Debugger))

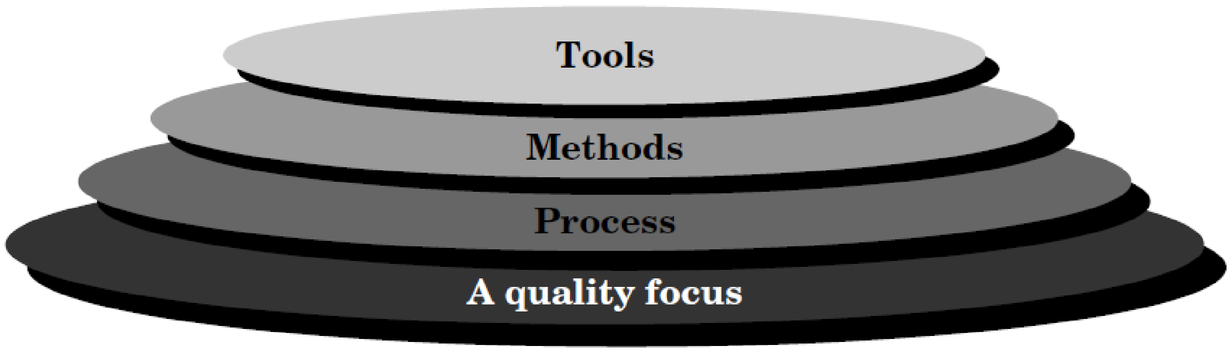
3. **Communication skill** (Verbal, Written, Presentation)

4. **Design Skill** (s/w engineer must be familiar with several application domain)

7.4.4 Factor in emergence of software engineering:

1. People who are developing software were consistently wrong in their estimation of time, effort and cost.
2. Reliability and maintainability was difficult of achieved
3. Fixing bug in a delivered software was difficult
4. Delivered software frequently didn't work
5. Changes in ration of hw to s/w cost
6. Increased cost of software maintenance
7. Increased demand of software
8. Increased demand for large and more complex software system
9. Increasing size of software

7.4.5 S/W ENGINEERING PRINCIPLES:-



Software engineering is a layered technology. The bedrock that supports software engineering is a quality focus. The foundation for software engineering is the ***process* layer**. Software engineering process is the glue that holds the technology layers together and enables rational and timely development of computer software. Process defines a framework for a set of ***key process areas*** that must be established for effective delivery of software engineering technology. The key process areas form the basis for management control of software projects and establish the context in which technical methods are applied, work product (models, documents, data, reports, forms, etc.) are produced, milestones are established, quality is ensured, and change is properly managed.

Software engineering *methods* provide the technical how-to's for building software. That encompass requirements analysis, design, program construction, testing, and support. Software engineering methods rely on a set of basic principles that govern each area of the technology and include modeling activities and other descriptive techniques.

Software engineering *tools* provide automated or semi-automated support for the process and the methods. When tools are integrated so that information created by one tool can be used by another, a system for the support of software development, called *computer-aided software engineering(CASE),* is established. CASE combines software, hardware, and a software engineering database (a repository containing important information about analysis, design, program construction, and testing) to create a software engineering environment analogous to CAD/CAE (computer-aided design/engineering) for hardware.

7.4.6 S/W CHARACTERISICS:-

Characteristics of a s/w can be easily distinguished as of from the h/w. Therefore after analyzing the facts we can write the key characteristics as follows:-

a) Most s/w s are custom built rather than assembled from existing components.

b) s/w is developed or engineered not manufactured.

c) s/w is flexible.

d) s/w does not wear out.

7.4.7 CAUSES AND SOLUTION FOR S/W CRISIS:-

Software engineering appears to be among the few options available to tackle the present software crisis. The expenses that organizations all around the world are incurring on software purchases compared to those on hardware purchases have been showing a worrying trend over the years

Organizations are spending larger and larger portions of their budget on software not only are the software products turning out to be more expensive than hardware, but also presented lots of other problems to the customers such as: software products are difficult to alter, debug, and enhance; use resources non optimally; often fail to meet the user requirements; are far from being reliable; frequently crash; and are often delivered late.

Other factors are larger problem sizes, lack of adequate training in software engineering, increasing skill shortage, and low productivity Improvements.

The s/w that mixes the specification must be produced.

i) s/w validation

The s/w must be validated to ensure that it performs desired customer activities.

ii) s/w evolution:-

The s/w must evolve to meet changing customer needs with time.

7.4.8 Generic attributes in a software process:

1. Understandability
2. Visibility
3. Reliability
4. Robustness
5. Adaptability
6. Rapidity

7. Maintainability
8. Supportability

7.4.9 Characteristic of a software process:

1. Understandability
2. Visibility
3. Reliability
4. Robustness
5. Adaptability
6. Rapidity
7. Maintainability
8. Supportability

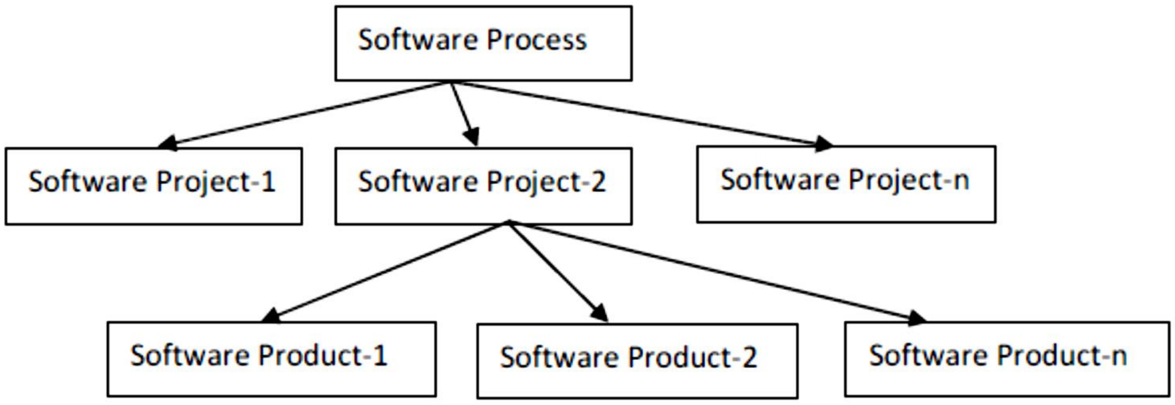
7.4.10 Software Project:

1. A project is a temporary endeavor undertaken to create a unique product.
2. Temporary means every project has a definite beginning and a definite end.
3. Unique means the product must be different in some ways from all similar Product

7.4.11 Software Product:

Outcome of software project is known as software product.

Software process vs software project vs software product



8.1 S/W LIFE CYCLE METHOD:-

8.1.1 Introduction:-

A software life cycle model (also called process model) is a descriptive and diagrammatic representation of the software life cycle. A life cycle model represents all the activities required to make a software product transit through its life cycle phases. It also captures the order in which these activities are to be undertaken.
In other words, a life cycle model maps the different activities performed to develop software product from its inception to retirement. Different life cycle models may map the basic development activities to phases in different ways.

The need for s/w life cycle model:-

- The development team must identify a suitable life cycle model for the particular project and then adhere to it.
- Without using of a particular life cycle model the development of a software product would not be in a systematic and disciplined manner. So when a software product is being developed by a team there must be a clear understanding among team members about when and what to do, Otherwise it would lead to chaos and project failure.
- A software life cycle model defines entry and exit criteria for every phase. A phase can start only if its phase-entry criteria have been satisfied. So without software life cycle model the entry and exit criteria for a phase cannot be recognized. Without software life cycle models (such as classical waterfall model, iterative waterfall model, prototyping model, evolutionary model, spiral model etc.) it becomes difficult for software project managers to monitor the progress of the project.

8.1.2 Different s/w life cycle model:-

Many life cycle models have been proposed so far. Each of them has some advantages as well as some disadvantages. A few important and commonly used life cycle models are as follows:

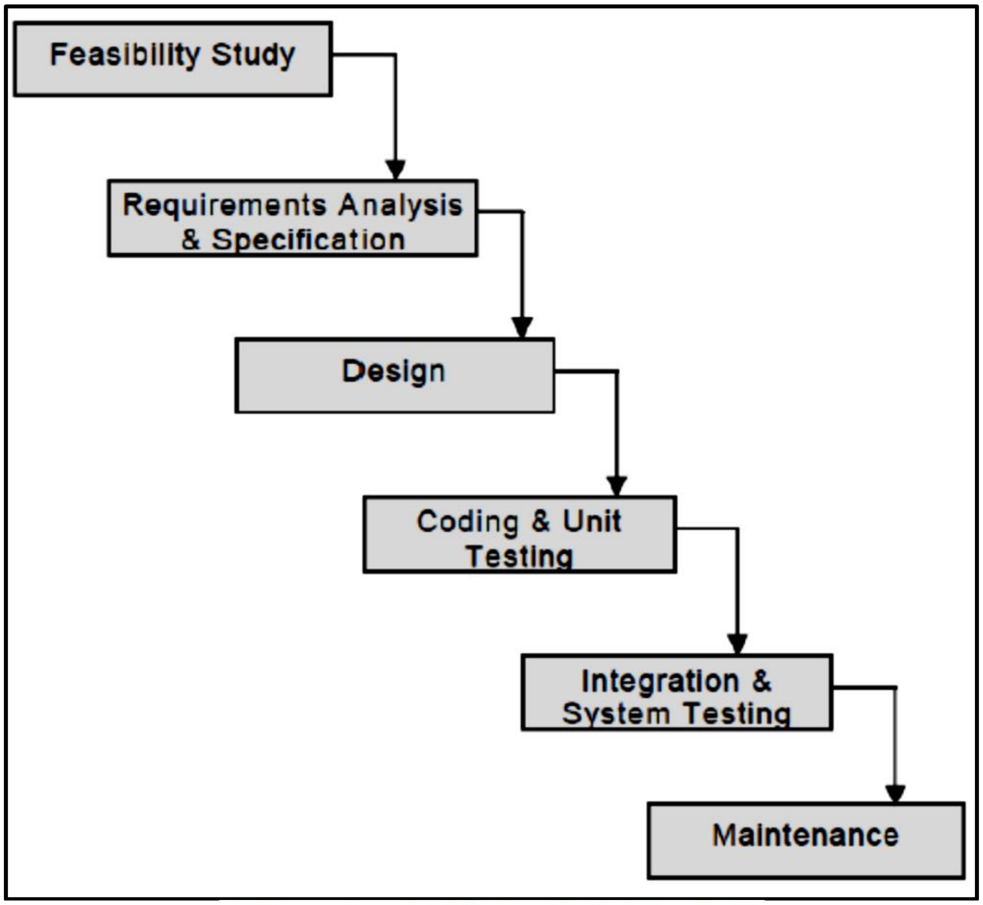
a.Classical Waterfall Model

b. Iterative Waterfall Model

c. Prototyping Model

d. Evolutionary Model

e.Spiral Model


Feasibility Study
Requirements Analysis & Specification
Design
Coding & Unit Testing
Integration & System Testing
Maintenance

Classical Waterfall Model

8.1.3 Activities in each phase of the life cycle:

- **Activities undertaken during feasibility study:**

The main aim of feasibility study is to determine whether it would be financially and technically feasible to develop the product. Rough understanding between the team members about estimate based on the client side requirement. After over all discussion they search for variety of solution on the basis of kind of resources and time requirement etc.

- **Activities undertaken during requirements analysis and specification:**

The aim of the requirements analysis and specification phase is to understand the exact requirements of the customer and to document them properly. This phase consists of two distinct activities, namely

- Requirements gathering and analysis, and
- Requirements specification

- **Activities undertaken during feasibility study: -**

The main aim of feasibility study is to determine whether it would be financially and technically feasible to develop the product. Rough understanding between the team members about estimate based on the client side requirement. After over all discussion they search for variety of solution on the basis of kind of resources and time requirement etc.

- **Activities undertaken during requirements analysis and specification:**

The aim of the requirements analysis and specification phase is to understand the exact requirements of the customer and to document them properly. This phase consists of two distinct activities, namely

- Requirements gathering and analysis, and
- Requirements specification

The goal of the requirements gathering activity is to collect all relevant information from the customer regarding the product to be developed. This is done to clearly understand the customer requirements so that incompleteness and inconsistencies are removed.

- After all ambiguities, inconsistencies, and incompleteness have been resolved and all the requirements properly understood, the requirements specification activity can start.
- During this activity, the user requirements are systematically organized into a Software Requirements Specification (SRS) document.

- **Activities undertaken during design: -**

The goal of the design phase is to transform the requirements specified in the SRS document into a structure that is suitable for implementation in some programming language. During the design phase the software architecture is derived from the SRS document. Two distinctly different approaches are available:
-the traditional design approach and
- the object-oriented design approach.

- **Activities undertaken during coding and unit testing:-**

The purpose of the coding and unit testing phase (sometimes called the implementation phase) of software development is to translate the software design into source code. Each component of the design is implemented as a program module. The end-product of this phase is a set of program modules that have been individually tested then proceeds for next stage. During this phase, each module is unit tested to determine the correct working of all the individual modules. It involves testing each module in isolation way as this is the most efficient way to debug the errors identified at this stage.

- **Activities undertaken during integration and system testing: -**

Integration of different modules is undertaken once they have been coded and unit tested. During the integration and system testing phase, the modules are integrated in a planned manner. Integration is normally carried out incrementally over a number of steps. During each integration step, the partially integrated system is tested and a set of previously planned modules are added to it. Finally, when 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 conforms to its requirements laid out in the SRS document. System testing usually consists of three different kinds of testing activities:

α – testing: It is the system testing performed by the development team.

β – testing: It is the system testing performed by a friendly set of customers.

*Under revision acceptance testing: It is the system testing performed by the customer himself after the product delivery to decide whether to accept or reject the delivered product.

- **Activities undertaken during maintenance: -**

Maintenance of a typical software product requires much more effort than the effort necessary to develop the product itself. Maintenance involves performing any one or more of the following three kinds of activities:

- Correcting errors that were not discovered during the product development phase. This is called corrective maintenance.
- Improving the implementation of the system, and enhancing the functionalities of the system according to the customer's requirements. It is called perfective maintenance.
- Porting the software to work in a new environment. For example, porting may be required to get the software to work on a new computer platform or with a new operating system. It is called adaptive maintenance.

8.1.4 Disadvantages of waterfall model:-

i) It can not handle satisfactorily different types of risks associated with real time s/w project, because, the requirements have to be pre decided by the client.

ii) Most real life project can't follow the exact frame sequence of the waterfall model.

8.2 S/W REQUIREMENT AND SPECIFICATION:-

The aim of the requirements analysis and specification phase is to understand the exact requirements of the customer and to document them properly. This phase consists of two distinct activities, namely

- Requirements gathering and analysis, and
- Requirements specification

a) Requirements gathering and analysis:-

The goal of the requirements gathering activity is to collect all relevant information from the customer regarding the product to be developed and analyzed.
This phase is done to clearly understand the customer requirements so that incompleteness and inconsistencies are removed. This activity begins by collecting all relevant data regarding the product to be developed from the user and from the customer through interviews and discussions.

For example, to perform the requirements analysis of a business accounting software required by an organization, the analyst might interview all the accountants of the organization to ascertain their requirements. The data collected from such a group of users usually contain several contradictions and ambiguities, since each user typically has only a partial and incomplete view of the system. Therefore it is necessary to identify all ambiguities and contradictions in the requirements phase and resolve them through further discussions with the customer.

b) Requirements specification:-

After all ambiguities, inconsistencies, and incompleteness have been resolved and all the requirements properly understood, the requirements specification activity can start. During this activity, the user requirements are systematically organized into a Software

Requirements Specification (SRS) document.

The customer requirements identified during the requirements gathering and analysis activity are organized into a SRS document.

The important components of this document are:

- functional requirements of the system
- nonfunctional requirements of the system
- goal of implementation.

Functional requirements:-

The functional requirements part discusses the functionalities required from the system. The system is considered to perform a set of high level functions $\{fi\}$. The functional view of the system is shown in fig.3.1. Each function fi of the system can be considered as a transformation of a set of input data (ii) to the corresponding set of output data (*oi*). The user can get some meaningful piece of work done using a highlevel function.

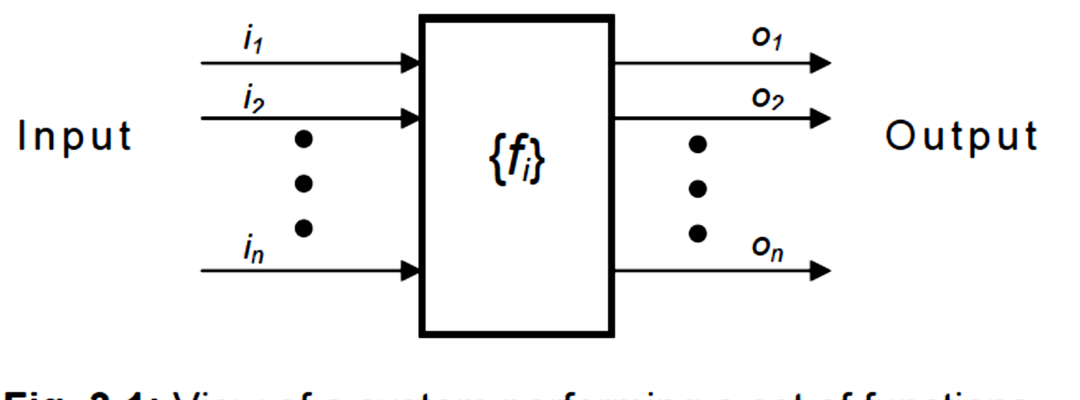


Fig. 3.1: View of a system performing a set of functions

Nonfunctional requirements:-

Nonfunctional requirements deal with the characteristics of the system which cannot be expressed as functions - such as the maintainability of the system, portability of the system, usability of the system, etc. It may include:

- reliability issues,
- accuracy of results,
- human - computer interface issues,
- constraints on the system implementation, etc.

8.3 Goals of implementation:-

The goals of implementation part documents, some general suggestions regarding development such as guide trade-off among design goals, document issues as revision to system functionalities that may required in future, new devices to be supported in the future, reusability etc.

These are the issues which the developers might keep in their mind during development so that the developed system may meet some aspects that are not required immediately.