


SOFTWARE ENGINEERING CONCEPTS



INTRODUCTION TO SE

- Software engineering is an engineering branch associated with development of software product using well-defined scientific principles, methods and procedures. The outcome of software engineering is an efficient and reliable software product.



- 
- Software project management has wider scope than software engineering process as it involves communication, pre and post delivery support etc.

COURSE OBJECTIVES

- At the end of the course should the student should be equipped with the understanding of software engineering concepts. That is
- **software product,**
- **software design and development process,**
- **software project management**
- **and related design complexities etc.**

SE OVERVIEW

- What is Software?
- What is Engineering?



What is Software?

- **Software** is more than just a program code.
- A **program** is an **executable code**, which serves some computational purpose.



What is Software?

- **Software** is considered a **collection** of
- **executable programming code**,
- associated **libraries** and **documentations**.
- when made for a **specific requirement** is called **software product**.



What is Software?

- **Software** when made up of **a specific requirement** is called a **software product**.



What is Engineering?

- **Engineering** is all about
- developing **products**,
- using well-defined, **scientific principles** and **methods**.
- We apply mathematical laws ...differential eqns, laplace, fourier to design filters, tuned circuits. Physics EM waves to design antenaes.
- Chemistry biology etc to design sensors eg blood pressure pulse rate stress levels.

WHAT IS SOFTWARE ENGINEERING?

- **Software engineering** is an engineering branch associated with **development of software product** using well-defined **scientific principles, methods and procedures**.
- The outcome of software engineering is an **efficient and reliable software product**.



SOFTWARE ENGINEERING CONCEPT



SOFTWARE ENGINEERING CONCEPT

- The above diagram provides an illustration of the stages that a software production process goes through.
- Imagine any engineering product does go from an idea on the drawing board that has to satisfy certain requirements. The product goes through analysis design and actual production.
- Initially the software product is a concept thus needs deeper analysis from abstraction to concrete realisation.

SOME SE DEFINITIONS- IEEE

- **1 The application of a systematic, disciplined, quantifiable approach to the development, operation and maintenance of software; that is, the application of engineering to software.**
- **2 The study of approaches as in the above statement.**

SOME SE DEFINITIONS- Fritz Bauer (German computer scientist)

- Software engineering is the Establishment and use of sound engineering principles to obtain economically software that is reliable and works on real machines efficiently.

SOFTWARE EVOLUTION



SOFTWARE EVOLUTION

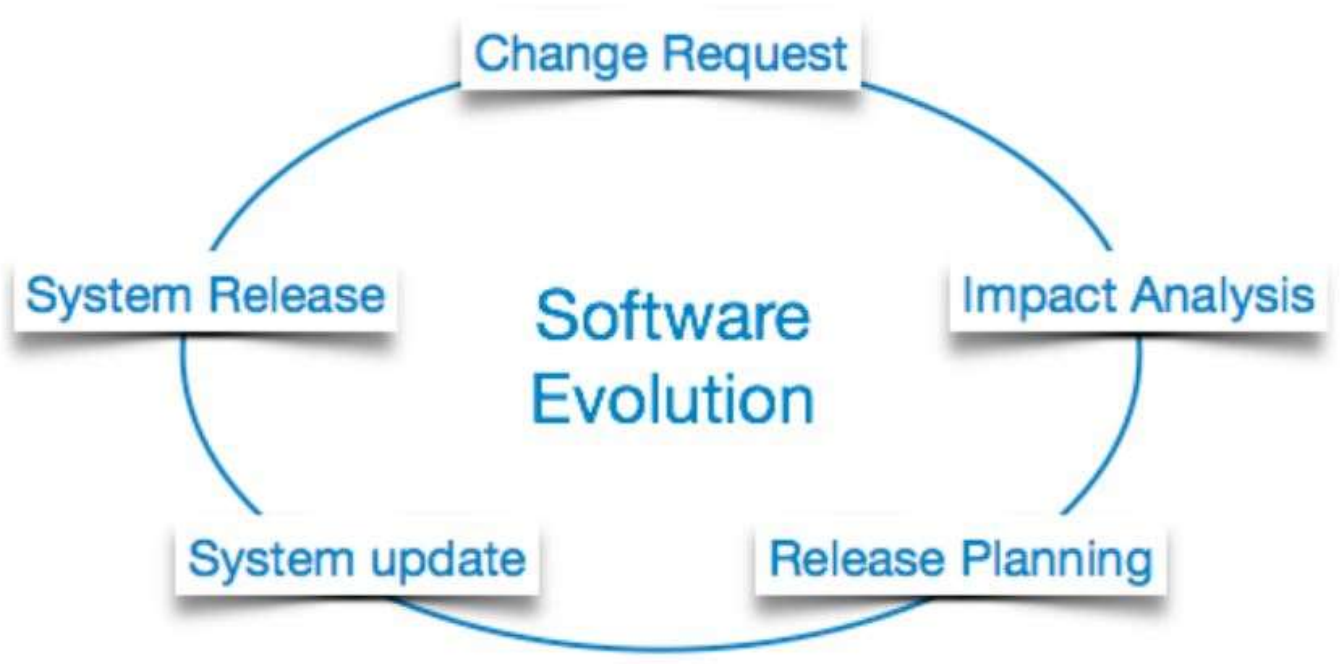
- The process of developing a software product using software engineering principles and methods is referred to as **software evolution**.



SOFTWARE EVOLUTION

- This includes the **initial development** of software and its **maintenance** and **updates**, till **desired software product** is developed, which **satisfies the expected requirements**






Software Evolution stages

- requirement gathering process.
- software prototype creation by developers
- users feedback at the early stage of software product development.
- The users suggest changes, on which several consecutive updates and maintenance keep on changing too.



- 
- This process changes to the original software, till the desired software is accomplished. Even after the user has desired software in hand, the advancing technology and the changing requirements force the software product to change accordingly

Special Note

- Re-creating software from scratch and to go one-on-one with requirement is not feasible. The only feasible and economical solution is to update the existing software so that it matches the latest requirements.



Software Evolution Laws

- Lehman has given laws for software evolution. He divided the software into **three different categories**:
 - ***S-type*** that is static-type
 - ***P-type*** practical -type
 - ***E-type*** embedded -type

Lehman's Static Software Evolution Law

- Lehman has given laws for software evolution. He divided the software into **three different**
- **categories:**
- **S-type** is a software, which works strictly according to defined specifications and solutions.
- The solution and the method to achieve it, both are immediately understood before coding.
- The s-type software is least subjected to changes hence this is the simplest of all.
- For example, **calculator program for mathematical computation.**

Lehman's Practical Software Evolution Law


- *P-type* is a software with a collection of procedures.
- This is defined by
 - exactly what procedures can do. In this software, the specifications can be described but the
 - solution is not obvious instantly. For example, **gaming software**.

Lehman's Embedded type Software Evolution

- *E-type or embedded*-type software works closely as the requirement of **real-world**
- **environment.**
- This software has a high degree of evolution as there are various changes in
- laws, taxes etc. in the real world situations. For example, **Online trading software.**

The Eight laws for E-Type software evolution



- 
- In order that the software evolution follow the real world. The following laws must be in place..

The Eight laws for E-Type software evolution


1 Continuing change – An E-type software system must evolve in a manner that it **continues to adapt** to the real world changes, else it becomes progressively less useful. *Redundant!*

2 Increasing complexity - As an E-type software system evolves, its complexity should continue to increase unless work is done to maintain or reduce it. (i.e. Some modification is done rendering it less complex)

The Eight laws for E-Type software evolution

3 Conservation of familiarity - The familiarity with the software or the **knowledge** about **how** and **why** was it developed in that particular manner etc. must **be retained** at any cost, **to implement the changes** in the system.

4 Continuing growth- In order for an E-type system to resolve some business problem, **its size** of implementing the changes **grows according to the lifestyle changes** of the **business**.



5 Reducing quality - The quality of an E-type system **will appear to be declining** unless it is rigorously maintained and adapted to operational environment change.

6 Feedback systems- E-type evolution processes constitute **multi-level, multi-loop, multi-agent** feedback systems and must be treated as such to achieve significant improvement over any reasonable base.

The Eight laws for E-Type software evolution

7 Self-regulation - E-type system evolution processes are self-regulating with the distribution

- of product and process measures close to normal.

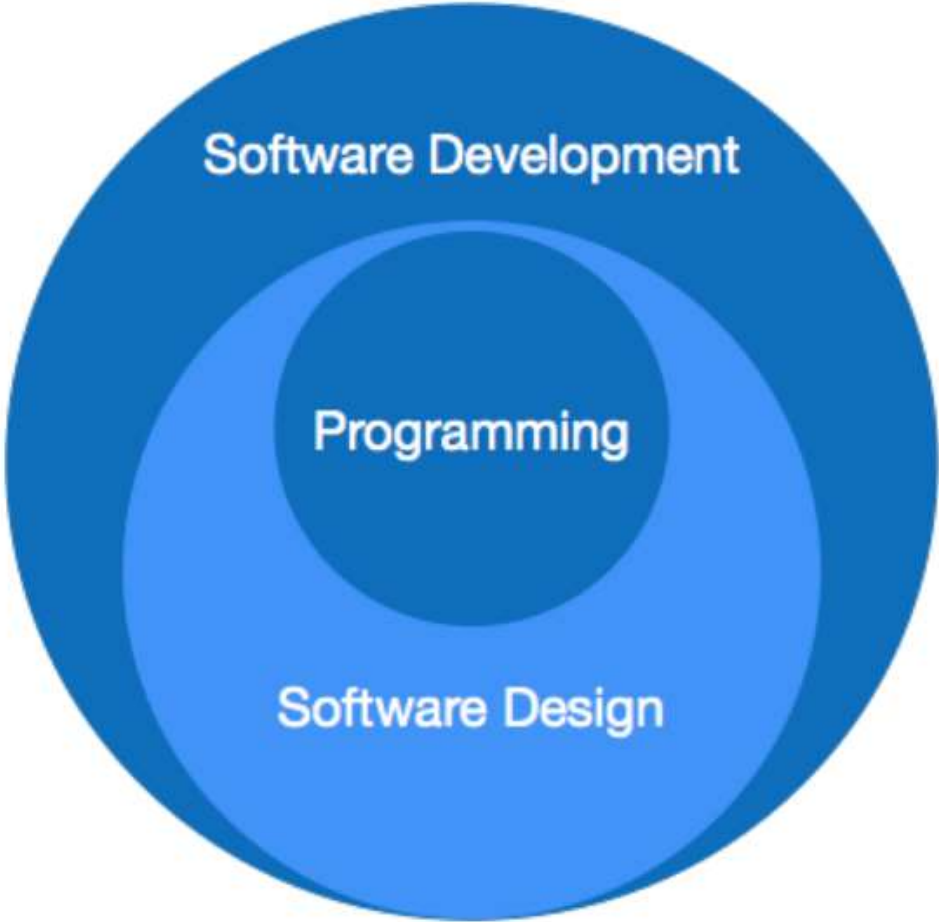
8 Conservation of Organizational stability(invariant work rate) - the average effective global activity rate in an evolving E-type system is invariant over the product's lifetime. (That is its activity is constant throughout its life.)


Software Paradigms



Software Paradigms

- Software paradigms refer to the **methods** and **steps**, which are **taken while designing the software**.
- There are many methods proposed which are at work today, there is the need to see where in the
- software engineering these paradigms stand. These can be combined into various categories,
- though each of them is contained in one another:



- 
- Programming paradigm is a subset of Software design paradigm which is further a subset of
 - Software development paradigm.

Software Development Paradigm

- This Paradigm is known as **software engineering paradigms** where all the engineering concepts pertaining to the development of software are applied.
- It includes various researches and requirement gathering which helps the software product to build. It consists of –
 - ***Requirement gathering***
 - ***Software design***
 - ***Programming***

Software Design Paradigm

- This paradigm is a part of Software Development and includes –
- *Design*
- *Maintenance*
- *Programming*



Programming Paradigm


- This paradigm is related closely to programming aspect of software development. This includes –
 - ***Coding***
 - ***Testing***
 - ***Integration***


NEED FOR SOFTWARE ENGINEERING



Need of Software Engineering


- The need of software engineering arises because of higher rate of change in user requirements
- and environment on which the software is working. The main factors that have resulted in the need of producing software products using structured methods are as follows
- **Large software** - It is easier to build a wall than a house or building, likewise, as the size
- of software becomes large engineering has to step in to give SW a scientific process.


- 
- **Scalability-** If the software process were not based on scientific and engineering concepts, it
 - would be easier to re-create new software than to scale an existing one. Larger systems evolve from small-scale ones...database from a few tables to 100s plus
 - **Cost-** As hardware industry has shown its skills and huge manufacturing has lower down the
 - price of computer and electronic hardware. But the cost of software remains high if proper processes are not adapted.

- 
- ***Dynamic Nature-*** The always growing and adapting nature of software hugely depends
 - upon the environment in which user works. If the nature of software is always changing, new
 - enhancements need to be done in the existing one. This is where software engineering plays
 - a good role.
 - ***Quality Management-*** Better process of software development provides better and quality
 - software product.

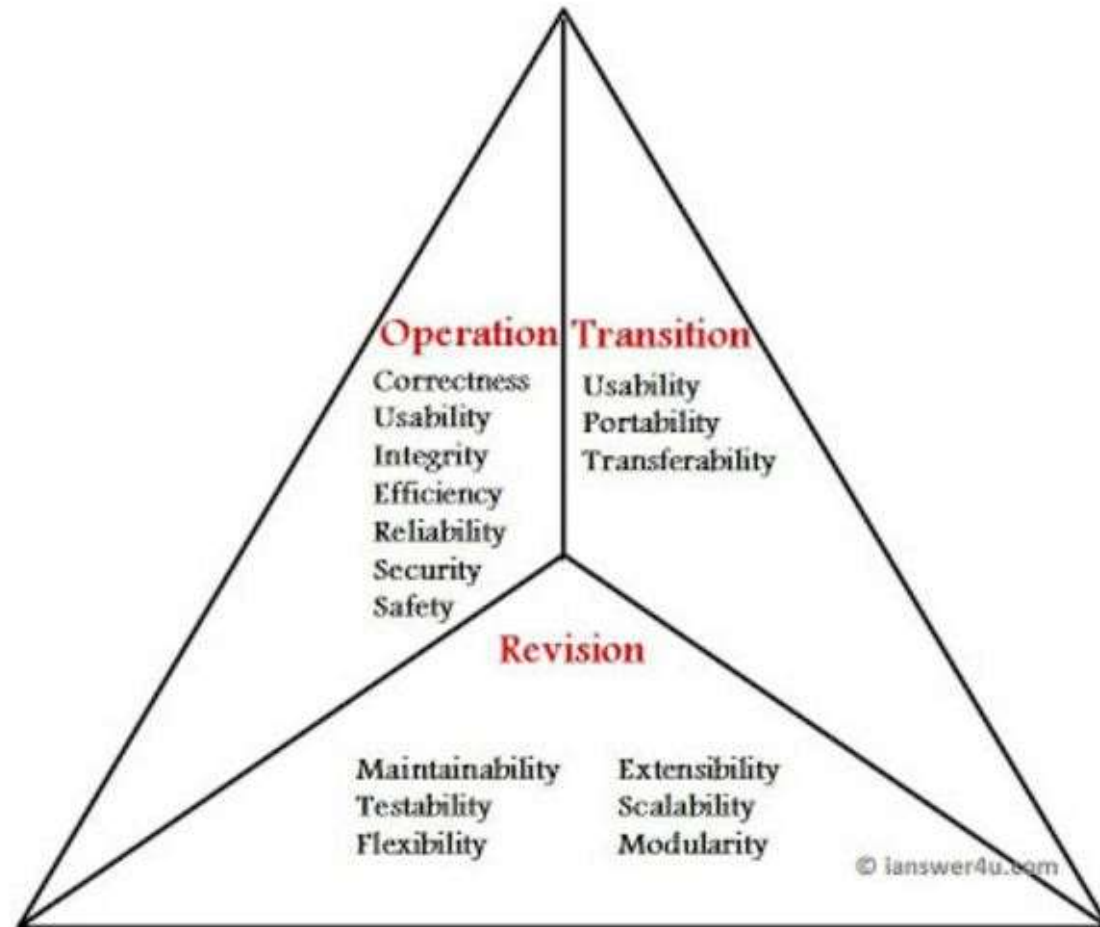
Characteristics of good software



- 
- First and foremost, a software product must **meet all the requirements of the customer or end-user.**
 - the **cost of developing** and **maintaining** the software should be low.
 - The development of software should be completed in the specified **time-frame.**

- 
- These are the obvious things which are expected from any project (and **software development** is a project in itself).
 - The three characteristics of good application software are :-
 - 1) **Operational Characteristics**
 - 2) **Transitional Characteristics**
 - 3) **Revision Characteristics**
 - These set of factors can be easily explained by Software Quality Triangle.

The Software Quality Triangle.



Software Operational Characteristics

- a) **Correctness:** The software should meet all the specifications stated by the customer. (ie answers the question ..Is it what the customer ordered?)

- b) **Usability/Learnability:** The amount of efforts or time required to learn how to use the software should be less. This makes the software **user-friendly** even for IT-illiterate people. (Naïve users...is there enough documentation..to start-run..operate the software...?)

SW Operational Characteristics

c) **Integrity** : Just like medicines have side-effects, in the same way a software may have a side-effect i.e. it may affect the working of another application. But a quality software should not have side effects.

d) **Reliability** : The software product should not have any defects. Not only this, it shouldn't fail while execution. (No hidden aspects...lives up to expectations no surprises during operation...bugs that may cause it to hang..crash etc)

SW Operational Characteristics

- e) **Efficiency** : This characteristic relates to the way software uses the available resources. The software should make effective use of the storage space and execute commands as per desired timing requirements. (Does not slow down or use up system resources eg memory. Is multithreaded can run sub- processes or threads to assist OS..)
-
- f) **Security** : With the increase in security threats nowadays, this factor is gaining importance. The software shouldn't have ill effects on data / hardware. Proper measures should be taken to keep data secure from external threats. (Product does not take over OS processes...is separated from host system...operates outside of the hardware,,eg uses virtual machine...software that emulates hardware without affecting it.)
-
- g) **Safety** : The software should not be hazardous to the environment/life.
(Some Software systems are used in patient monitoring, emergency relief, Accident fire services. They should be able to perform safely without harming the users..eg not provide false alarms ..eg fire where there is none or wrong patient health readings eg in a neonatal incubator..humidifiers..temperature etc. wrong location or warning of an earthquake...etc)

SW Transition Characteristics

- a) **Interoperability** : Interoperability is the ability of software to exchange information with other applications and make use of information transparently.
- b) **Reusability** : If we are able to use the software code with some modifications for different purpose then we call software to be reusable. (Object Oriented programming OOP uses classes which can be “inherited” . These present a core program that a user can modify eg the MSOffice Document is a class which a user can use without having to rewrite it.....a user can add-on contents. Most classes at lower levels can be implemented in code..eg Java)
- c) **Portability** : software portability is demonstrated by its ability of to perform same functions across all environments and platforms. Same software being used on different machines/hardware. Same software useable with different OSes.

SW Revision Characteristics

- a) **Maintainability** : Any kind of user should be able to carry out maintenance of the software with ease.
- b) **Flexibility** : Changes in the software should be easy to make.
- c) **Extensibility** : It should be easy to increase or extend the functions performed by the SW.

SW Revision Characteristics

- - d) **Scalability** : It should be very easy to upgrade it for more work-load (or for an increased number of users).
 - e) **Testability** : Testing the software should be easy and comprehensive.(Easy especially if its modularised...)

SW Revision Characteristics

- f) **Modularity** : Any software is said to be made of units and modules which are independent of each other. (These modules are then integrated to make the final software. If the software is divided into separate independent parts that can be modified, **tested separately**, it has high modularity. High modularity implies loose coupling. Programming uses functions...methods..subroutines to ensure modularity)

SUMMARY ON GOOD SW CHARACTERISTICS

- Importance of any of these factors varies from application to application. In systems where **human life** is at stake – critical systems eg life support, fire, emergency, relief accident- **integrity** and **reliability factors** must be given prime importance.
- In any **business related application** where cost saving is primary **usability** and **maintainability** are key factors to be considered.

SUMMARY ON GOOD SW CHARACTERISTICS

- Always remember in Software Engineering, **quality of software is everything**,
- Thus the developer/ Engineer must try to deliver a product which has **all these characteristics and qualities at ALL COST!!**



Conclusion on SE intro

In short, Software engineering is a branch of computer science, which uses **well-defined engineering concepts** required to produce

- **efficient,**
- **durable,**
- **scalable,**
- **in-budget and on-time** software products.