Tribhuvan University
Institute of Engineering
Pulchowk Campus
Department of Electronics and Computer Engineering

Software Engineering

[Subject Code: CT 601]

by
Er. Bishwas Pokharel
Lecturer, Kathford Int'l College of Engg. and Mgmt.

Date: 27th Dec, 2017

Overall Course Outline:

SOFTWARE ENGINEERING

CT 601

Lecture : 3 Year : III

Tutorial : 1 Part : I

Practical : 1.5

Course Objectives:

This course provides a systematic approach towards planning, development, implementation and maintenance of system, also help developing software

projects.

1. Software Process and requirements (12 hours)

1.1. Software crisis

1.2. Software characteristics

1.3. Software quality attributes

1.4. Software process model

1.5. Process iteration

1.6. process activities

1.7. Computer-aided software engineering

1.8. Functional and non –functional requirements

1.9. User requirements

1.10. System requirement

1.11. Interface specification

1.12. The software requirements documents

1.13. Feasibility study

1.14. Requirements elicitation and analysis

1.15. Requirements validation and management

2. System models (3 hours)

2.1. Context models

2.2. Behavioural models

2.3. Data and object models

3. Architectural design (6 hours)

3.1. Architectural design decisions

3.2. System organization

3.3. Modular decomposition styles

3.4. Control styles

3.5. Reference architectures

3.6. Multiprocessor architecture

3.7. Client –server architectures

3.8. Distributed object architectures

3.9. Inter-organizational distributed computing

4. Real –time software design (3 hours)

4.1. System design

4.2. Real-time operating systems

4.3. Monitoring and control systems

4.4. Data acquisition systems

5. Software Reuse (3 hours)

5.1. The reuse landscape

5.2. Design patterns

5.3. Generator –based reuse

5.4. Application frameworks

5.5. Application system reuse

6. Component-based software engineering (2 hours)

6.1. Components and components models

6.2. The CBSE process

6.3. Component composition

7. Verification and validation (3 hours)

7.1. Planning verification and validation

7.2. Software inspections

7.3. Verification and formal methods

7.4. Critical System verification and validation

8. Software Testing and cost Estimation (4 hours)

8.1. System testing

8.2. Component testing

8.3. Test case design

8.4. Test automation

8.5. Metrics for testing

8.6. Software productivity

8.7. Estimation techniques

8.8. Algorithmic cost modeling

8.9. Project duration and staffingf

9. Quality management (5 hours)

9.1. Quality concepts

9.2. Software quality assurance

9.3. Software reviews

9.4. Formal technical reviews

9.5. Formal approaches to SQA

9.6. Statistical software quality assurance

9.7. Software reliability

9.8. A framework for software metrics

9.9. Matrices for analysis and design model

9.10. ISO standards

9.11. CMMI

9.12. SQA plan

9.13. Software certification

10. Configuration Management (2 hours)

10.1. Configuration management planning

10.2. Change management

10.3. Version and release management

10.4. System building

10.5. CASE tools for configuration management

Practical

The laboratory exercises shall include projects on requirements, analysis and designing of software system. Choice of project depend upon teacher and student, case studies shall be included too. Guest lecture from software industry in the practical session.

References:

1. Ian Sommerville, Software Engineering , Latest edition
2. Roger S. Pressman, Software Engineering –A Practitioner's Approach, Latest edition
3. Pankaj Jalote, Software Engineering-A precise approach, Latest edition
4. Rajib Mall, Fundamental of Software Engineering, Latest edition

Evaluation Scheme:

The questions will cover all the chapters in syllabus. The evaluation scheme will be as indicated in the table below:

Chapters	Hours	Marks distribution*
1	12	20
2	3	5
3	6	10
4	3	5
5	3	5
6	2	3
7	5	10
8	4	8
9	5	10
10	2	4
Total	45	80

*There may be minor deviation in marks distribution

Tribhuvan University
Institute of Engineering
Pulchowk Campus
Department of Electronics and Computer Engineering

Software Engineering

Chapter One

Software Process and requirements

by
Er. Bishwas Pokharel
Lecturer, Kathford Int'l College of Engg. and Mgmt.

Date: 27th Dec, 2017

Chapter One: **Software Process and requirements**

Course Outline: **12 hours, 20 Marks**

1.1. Software crisis

1.2. Software characteristics

1.3. Software quality attributes

1.4. Software process model

1.5. Process iteration

1.6. process activities

1.7. Computer-aided software engineering

1.8. Functional and non –functional requirements

1.9. User requirements

1.10. System requirement

1.11. Interface specification

1.12. The software requirements documents

1.13. Feasibility study

1.14. Requirements elicitation and analysis

1.15. Requirements validation and management

What is Software? Computer software is the product that software professionals build and then support over the long term.

What is the work product? From the point of view of a software engineer, the work product is the set of programs, content (data), and other work products that are computer software. But from the user's viewpoint, the work product is the resultant information that somehow makes the user's world better.

How should we define *software?*

Software is:

(1) instructions (computer programs) that when executed provide desired features, function, and performance;

(2) data structures that enable the programs to adequately manipulate information, and

(3) descriptive information in both hard copy and virtual forms that describes the operation and use of the programs.

Why is it important?

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

Software is classified according to the range of potential applications. These classifications are listed below:

1. **System software:** This class of software is responsible for managing and controlling operations of a computer system. System software is a group of programs rather than one program and is responsible for using computer resources efficiently and effectively. For example, operating system is system software, which controls the hardware, manages memory and multi-tasking functions, and acts as an interface between applications programs and the computer.

2.Real-time software: This class of software observes, analyses, and controls real world events as they occur. Generally, a real-time system guarantees a response to an external event within a specified period of time. For example, real-time software is used for navigation in which the computer must react to a steady flow of new information without interruption. Most of the defense organizations all over the world use real-time software to control their military hardware.

3. **Business software:** This class of software is widely used in areas where the management and control of financial activities is of utmost importance. The fundamental component of a business software comprises of payroll, inventory, accounting, and software that permits user to access relevant data from the database. These activities are usually performed with the help of specialized business software that facilitates efficient framework in the business operation and in management decisions

4. **Engineering and scientific software:** This class of software has emerged as a powerful tool to provide help in the research and development of next generation technology.

Applications, such as study of celestial bodies, study of under-surface activities, and programming of orbital path for space shuttle are heavily dependent on engineering and scientific software. This software is designed to perform precise calculations on complex numerical data that are obtained during real-time environment.

R, Matlab, IDE..

5. **Web-based software**: This class of software acts as an interface between the user and the Internet. Data on the Internet can be in the form of text, audio, or video format, linked with hyperlinks. Web browser is a web-based software that retrieves web pages from the Internet.
6. **Personal computer (PC) software:** This class of software is used for official and personal use on daily basis. The personal computer software market has grown over the last two decades from normal text editor to word processor and from simple paintbrush to advance image-editing software, database management system, financial accounting package, or a multimedia based software.

Software Characteristics

1. Software is developed or engineered, not manufactured in the classical sense

2. Software doesn’t “wear out”

3. Most software is custom-built, rather than being assembled from existing components

Continued…

1. Software is developed or engineered, not manufactured in the classical sense
   - Although similarities between H/W & S/W
   - **Similarities**
     - High quality needs to be achieved in both
     - Both depend on people &
     - Requires construction of product

Continued…

Differences

- In H/W quality problems are almost non existent /easily rectified in H/w but in S/W difficult
- Relationship between people & work achieved are different
- Product Construction approaches are different

– Software costs are concentrated in Engineering

2. Software doesn't "wear out"

(H/W failure)

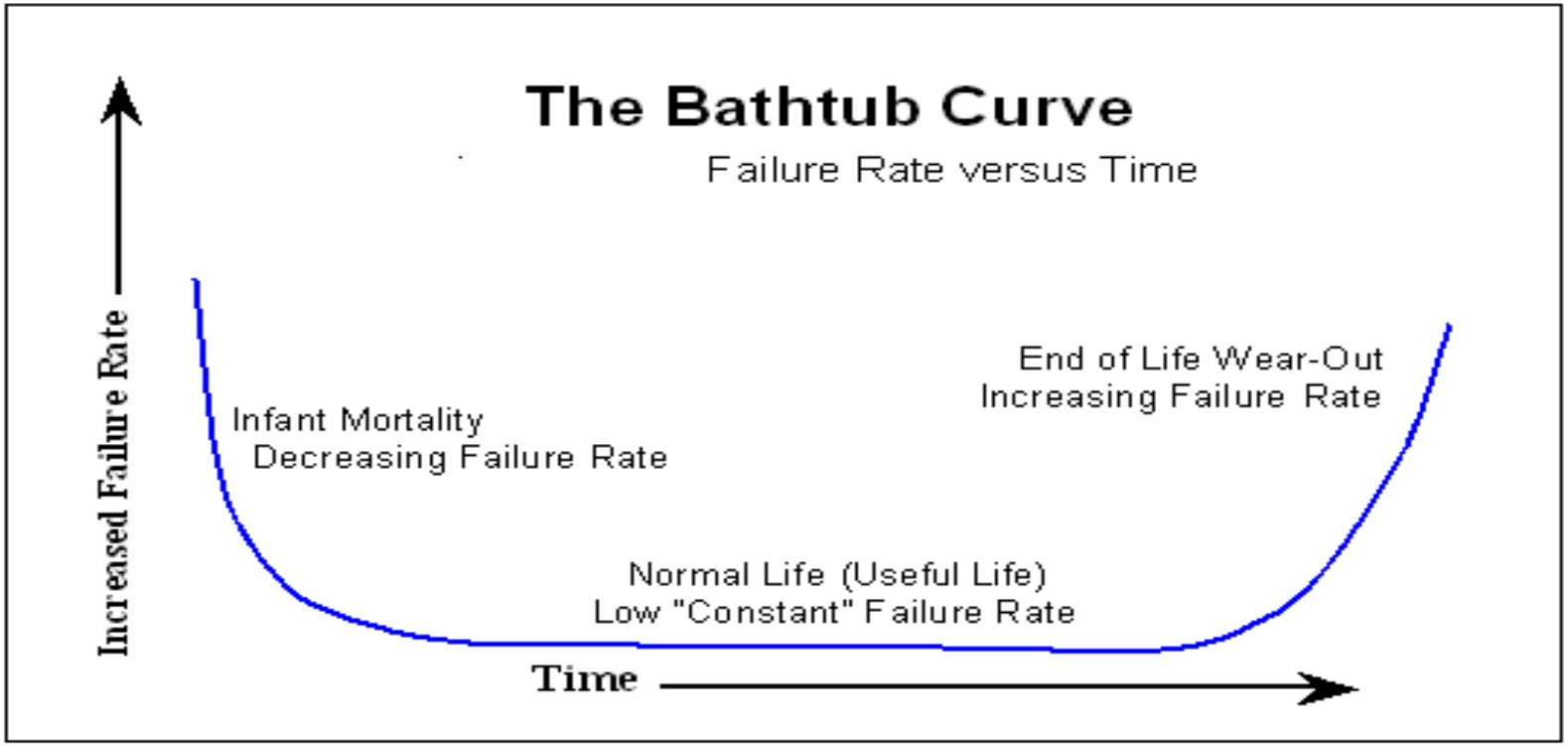


Figure 1 : Bathtub Curve

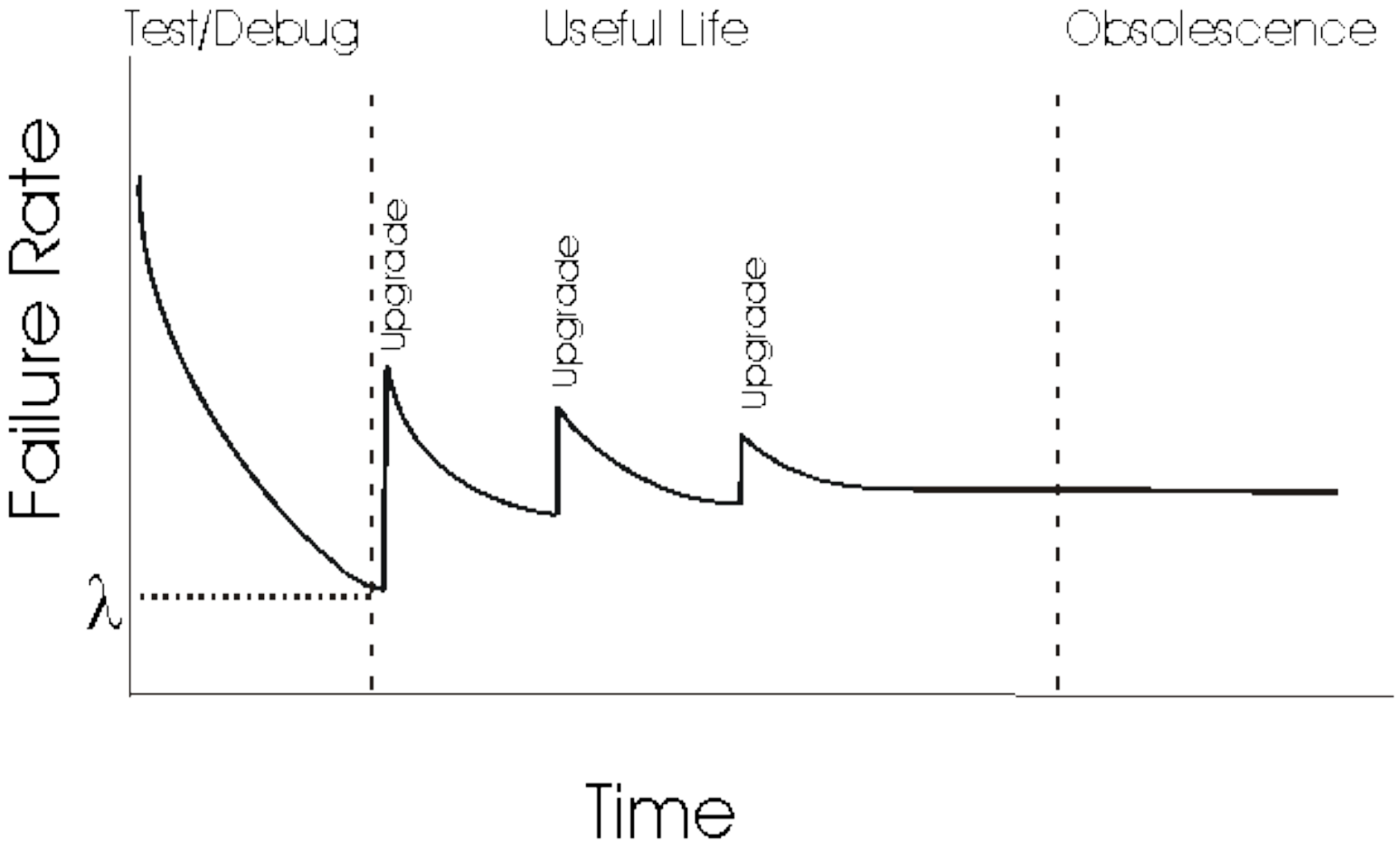
Figure 1 depicts failure rate vs time for hardware called bath tub curve

- The relationship, often called the "bathtub curve,“
- indicates that hardware exhibits relatively high failure rates early in its life (failures due to design /manufacturing defects);
- defects are corrected and the failure rate drops to a steady-state level (ideally, quite low) for some period of time.

Continued…

- As time passes
  - the failure rate rises again as hardware components suffer from the cumulative affects of dust, vibration, abuse, temperature extremes, and many other environmental maladies.
- Stated simply, the hardware begins to wear out.

S/W Failure


Test/Debug
Useful Life
Obsolescence
Failure Rate
λ
Upgrade
Upgrade
Upgrade
Time

Continued…

- Software is not susceptible to environmental maladies
- In theory, s/w should take the form of "idealized curve"
- However, Undiscovered defects in the beginning will cause high failure rates
- These are corrected (ideally, without introducing other errors) and the curve flattens as shown
- During the life time it undergo changes

Continued…

- Software is not susceptible to environmental maladies
- In theory, s/w should take the form of "idealized curve"
- However, Undiscovered defects in the beginning will cause high failure rates
- These are corrected (ideally, without introducing other errors) and the curve flattens as shown
- During the life time it undergo changes

Continued…

- it is likely that some new defects will be introduced, causing the failure rate curve to spike as shown
- Before the curve can return to the original steady-state failure rate, another change is requested, causing the curve to spike again
- Slowly, the minimum failure rate level begins to rise—the software is deteriorating due to change.

Continued…

- Another aspect of wear illustrates the difference between hardware and software.
- When a hardware component wears out, it is replaced by a spare part.
- There are no software spare parts.
- Every software failure indicates an error in design or in the process through which design was translated into machine executable code.
- Therefore, software maintenance involves considerably more complexity than hardware maintenance.

Continued…

3. Although the industry is moving toward component-based assembly, most software continues to be custom built.

- Consider the manner in which the control hardware for a computer-based product is designed and built.
- The design engineer draws a simple schematic of the digital circuitry, does some fundamental analysis to assure that proper function will be achieved, and

Continued…

- then goes to the shelf where catalogs of digital components exist.
- Each integrated circuit (called an *IC* or a *chip*) has a part number, a defined and validated function, a well-defined interface, and a standard set of integration guidelines.
- After each component is selected, it can be ordered off the shelf.

Continued…

- As an engineering discipline evolves, a collection of standard design components is created.
- Standard screws and off-the-shelf integrated circuits are only two of thousands of standard components that are used by mechanical and electrical engineers as they design new systems.

Continued…

- The reusable components have been created so that the engineer can concentrate on the truly innovative elements of a design, that is, the parts of the design that represent something new.
- In the hardware world, component reuse is a natural part of the engineering process.
- In the software world, it is something that has only begun to be achieved on a broad scale.

Continued…

- A software component should be designed and implemented so that it can be reused in many different programs.
- Today, we have extended our view of reuse to encompass not only algorithms but also data structure.
- Modern reusable components encapsulate both data and the processing applied to the data, enabling the software engineer to create new applications from reusable parts.

What are the factors with software quality?

- **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 software requirements specification(SRS) document have been correctly implemented.
- **Maintainability**: A software product is maintainable, if errors can be easily corrected, new functions can be easily added the product, and the functionalities of the product can be easily modified, etc

- **Functionality:** Refers to the degree of performance of the software against its intended purpose.
- **Reliability:** Refers to the ability of software to perform a required function under given conditions for a specified period.

But in real life there is not always guarantee to have software quality and poor software quality may arises due to the following reasons:

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

Poor Quality Software results errors and errors have caused large-scale financial losses as well as inconvenience to many, in addition it affect economic, political, and administrative system of various countries around the world. This situation where catastrophic failures have occurred is known as the **Software crisis.**

What is Software Crisis ?

The difficulty of writing the code for a computer program which is correct and understandable is referred to as software crisis.

or

Software crisis is also referred to as inability to hire enough qualified programmers.


Increasing Demand
Increasing Complexity
Increasing Challenges
Software Crisis
Same Workforce
Same Methods
Same Tools

- Software market today has a turnover of more than millions of rupees.
- Out of this, approximately 30% of software is used for personal computers and the remaining software is developed for specific users or organizations.
- Application areas, such as the banking sector are completely dependent on software application for their working. Software failures in these technology-oriented areas have led to considerable loss in terms of time, money, and even human lives.

History has seen many such failures. Some of these are listed below:

1) **1991 during Gulf War**: The USA use patriot missiles as a defense against Iraqi scud missile. However, patriot failed to hit the scud many times with cost life of 28 USA soldiers. **In an inquiry it is found that a small bug had resulted in miscalculation of missile path.**

2) **Arian- 5 Space Rocket:** In 1996, developed at cost of **$7000 Million Dollars** over a period of 10 years was destroyed within less than 1 minutes after its launch. As there was software bugs in rocket guidance system.

3) **"Dollar 924 lakhs":** In 1996, US bank credit accounts of nearly 800 customer with dollar 924 lakhs. This problem was due to main programming bug in the banking system.

4) **The North East blackout in 2003-** has been major power system failures in the history of north which involves **100 power plants, 50 million customer faced problem, $ 6 million dollar financial loss.**

5) **Year 2000(Y2k)** refers to the widespread snags in processing date after the Year 2000. In 1960-80 when shortened the **four digit date format** like 1972 to a **2 digit format** like 72 because of **"Limited Memory.** Because of that 2000 was shortened to 00. Million dollar were spent to handle this problem.

6) In June 1980, the North American Aerospace Defense Command (NORAD) reported that the US was under missile attack. The report was traced to a faulty computer circuit that generated incorrect signals. If the developers of the software responsible for processing these signals had taken into account the possibility that the circuit could fail, the false alert might not have occurred

Why study software engineering?

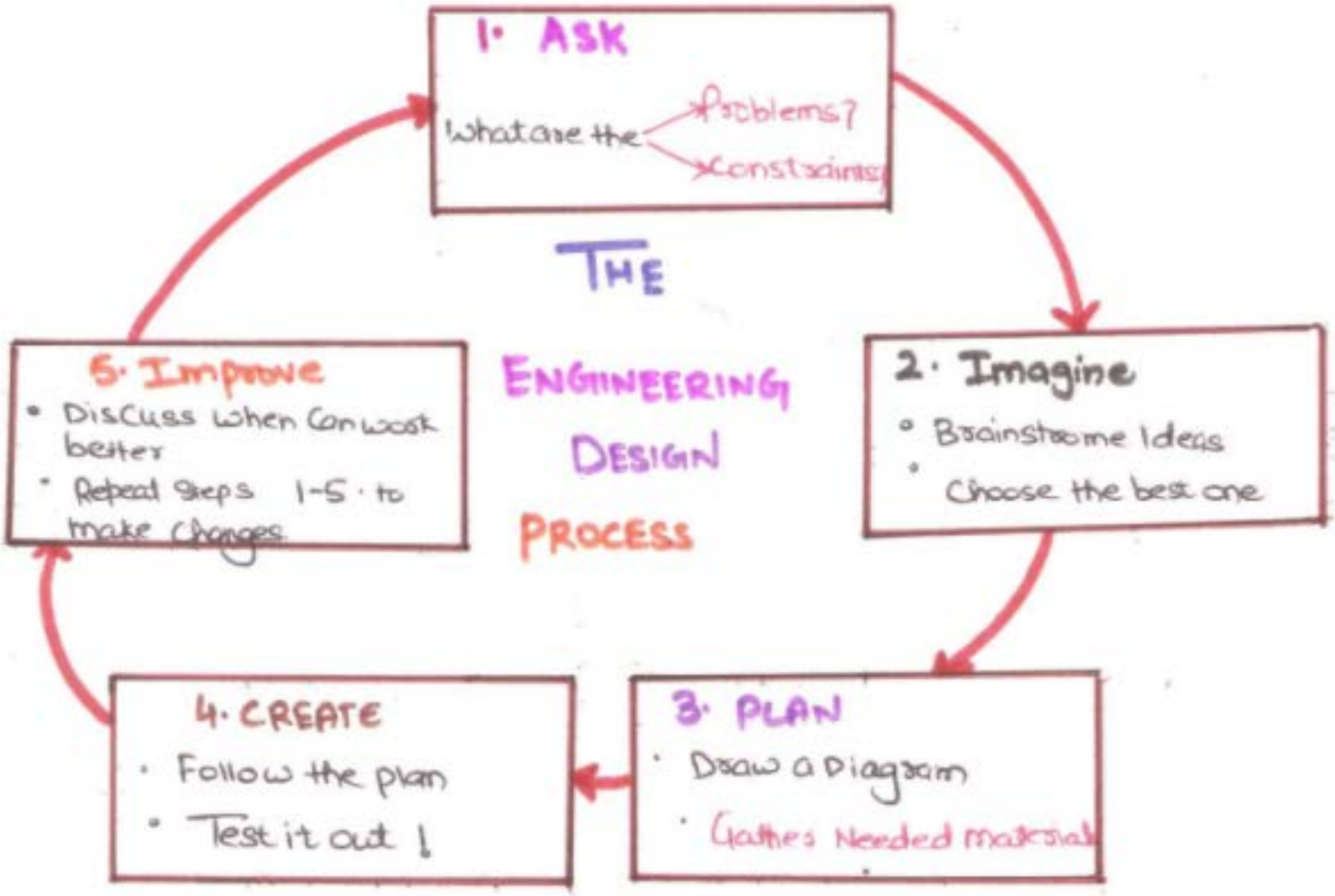
1) Higher productivity.
2) To acquire skills to develop large programs.
3) Ability to solve complex programming problems.
4) Learn techniques of specification design.
5) Better quality programmers.

What is Engineering?

Engineering is all about developing products, using well defined, scientific methods and principles.

OR,

Engineering is the application of scientific knowledge to solve the problem in Real world.


1. ASK
What are the problems?
constraints?
2. Imagine
Brainstorme Ideas
Choose the best one
3. PLAN
Draw a Diagram
Gather Needed materials
4. CREATE
Follow the plan
Test it out !
5. Improve
Discuss when Can work better
Repeat steps 1-5 to make changes
THE
ENGINEERING
DESIGN
PROCESS

What is software engineering?

It is an Engineering branch associated with the development of software product using well-defined scientific principles, method, and procedure. The outcome of software engineering is an efficient and reliable software product.

Definition of Software Engineering

IEEE Definition:

The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software

Another Definition of Software Engineering

The practical application of scientific knowledge in the design and construction of computer programs and the associated documentation required to develop, operate, and maintain them. (Boehm).

Objectives of Software Engineering

- To improve quality of software products
- To increase customer satisfaction
- To increase productivity
- To increase job satisfaction

Software engineering is not programming. Programming is an important part of software engineering.

“This is not a programming course”

Who takes part in Software Engineering?

A software engineer is an individual responsible for analysis, design, testing, implementation, and maintenance of effective and efficient software system. Generally, software engineer should possess the following qualities:

Problem solving skills:

Software engineer should develop algorithms and solve programming problems.

- **Programming_skills:** Software engineer should be well versed in data structures and algorithms, and must be expert in one or more programming languages and possess strong programming capabilities.

- **Design approaches:** Software engineer should be familiar with numerous design approaches required during the development of software.

Ex: Function oriented , Object oriented…

- **Software technologies:** Software engineer should have good understanding of software technologies. Ability to move among several levels of abstractions at different stages of the software project, from specific application procedures and requirements to the detailed coding level is also required.

Ex: Machine learning,Virtual reality, Internet of things and smart home tech..

- **Project management:** Software engineer should know how to make a project work, on time and on budget, in order to produce quality applications and systems.
- **Model of the application:** Software engineer should be able to create and use a model of the application to guide choices of the many tradeoffs that will be faced by him. The model is used to find answers to questions about the behavior of the system.

Ex: waterfall, prototype, agile…

Thank You!!!