Unit 1 – Software Engineering

Course Outcomes

Pre-requisites - Helpful

- Knowledge of one or more programming language
- Experience in the implementation of at least one software project/mini project

Course Outcomes [1/3]

- Understand Software Development Life Cycle (SDLC) models and the importance of *engineering software*.
- Creation of efficient Software Requirement Specification (SRS) documents to obtain the desired software product.
- Compare various software development methodologies and their applicability in developing a specific type of software product.

Unit 1 – Outcomes [2/3]

- Construct an efficient design specification document to obtain the desired product per user requirements.
- Be able to develop software applications using the concepts applicable to various phases of the software development life cycle.
- Study various software testing techniques and identify their relevance in developing quality software.

Unit 1 – Outcomes [3/3]

- Recognize software maintenance as a major activity in SDLC
- Examine the various tools used in all phases of Software construction
- Understand concepts of software project management and quality assurance

Introduction to Software Engineering

Outcomes & Introduction

Outcomes

- Acknowledge Pervasiveness of Software and recognize its Complexity
- Discuss Terminology and concepts required to understand Software Engineering
- Origin of and the need for Software Engineering

Introduction

- The modern world is unimaginable without software.
- Software has its presence across various industries.
- Software is pervasive in manufacturing, utilities, distribution, financial, education, and entertainment among numerous sectors.

Nature of Software [1/2]

- Software systems lack physical constraints.
- Quickly can become extremely complex to create, difficult to understand, and expensive to change.
- It is still an extremely challenging and complex endeavour to develop and maintain software applications.

Nature of Software [2/2]

- Software applications are getting larger, more complex, the demands change.
- Systems have to be built and delivered more quickly
- In many places ad-hoc approach to software development is prevalent.
- Therefore software tends to be unreliable, expensive to maintain and use.

Necessity of Engineering

- Engineering approach to develop and manage software is required.
- This entails education and training in software engineering methods.
- Thanks to software engineering achievements like space travel and the Internet, eCommerce, banking and other services we take for granted today are possible.

Introduction to Software Engineering

Define Software Engineering

Terminology

Software comprises of

- Instructions (computer programs) that when executed provide desired features, function, and performance;
- Data structures that enable the programs to adequately store and manipulate information and
- Documentation that describes the operation and use of the programs.
- Custom/Bespoke
- Off the Shelf / Generic

Engineering Defined

- The study of using scientific principles to design and build machines, structures, and other things, including bridges, roads, vehicles, and buildings: (American Dictionary)
- The art or science of making practical application of the knowledge of pure sciences, as physics or chemistry, as in the construction of engines, bridges, buildings, mines, ships, and chemical plants. (dictionary.com)

Software Engineering

IEEE fully defines software engineering as:

- The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software.
- [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.

Objectives of Software Engineering

- Engineering is about quality within schedule and budget.
- Software Engineering is a disciplined and systematic approach to develop quality software,
- Software that is delivered on time, within budget, and that satisfies its requirements.
- It relies on past experiences for techniques, methodologies, guidelines.

Engineering the Software [1/2]

- Concerned from system specification to maintaining the system
- Engineers apply theories, methods, and tools where these are appropriate.
- Discover solutions to problems even when there are no applicable theories and methods.

Engineering the Software [2/2]

- Solutions within organizational and financial constraints
- Not just technical processes of software development.
- Includes activities such as software project management …
- The development of tools, methods, and theories to support software production & maintenance.

Unit 1 – Software Engineering

Why Study Software Engineering ?

Try to answer these Questions ?

How would you build a temporary shelter for your pet dog?

PAUSE

Now answer these Questions ?

- **How would you go about building a bridge across river Ganga in Rishikesh?**
- **How would you build a Home for a large human Family of 20 members?**
- **Program to calculate the area of a triangle versus the problem of automating your College Operations**

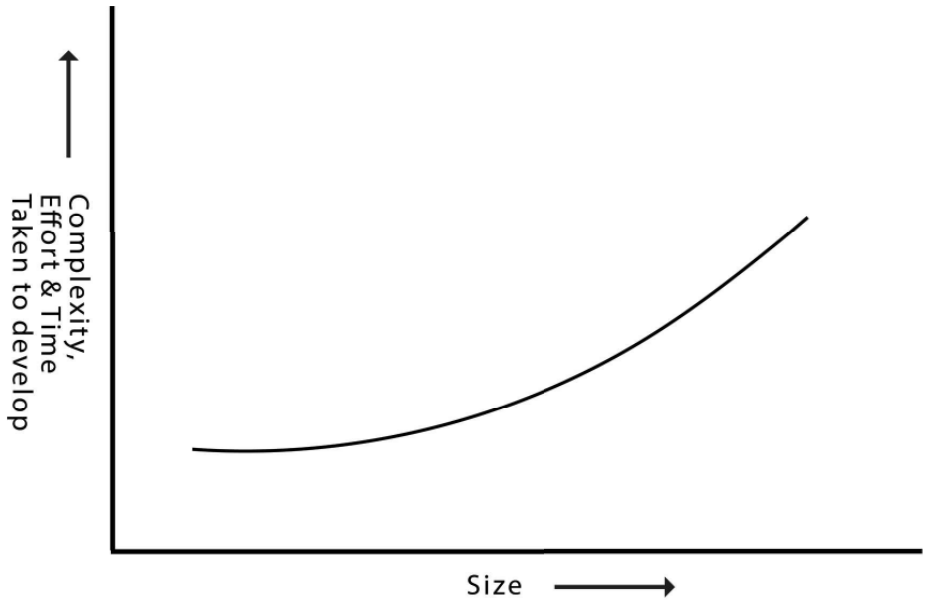
Need for Software Engineering!

- Academic journey, you are unable to appreciate need for Software Engineering
- Large software application, then software engineering principles are indispensable to achieve good quality software cost-effectively

Some Reasons for Studying SE[1/2]

- The ad hoc approach breaks down when the size of software increases.
- To acquire skills to develop large programs and work in large real-life projects.
- The exponential growth in complexity and difficulty level increases with code and project size.

Code Size vs. Complexity



Some Reasons for Studying SE[2/2]

- A program with 10,000 LOC is not just 10 times more difficult to develop, but may well turn out to be 100 times more difficult
- Understand how to break large projects into smaller and manageable parts

Unit 1 – Software Engineering

Origin of the Term Software Engineering

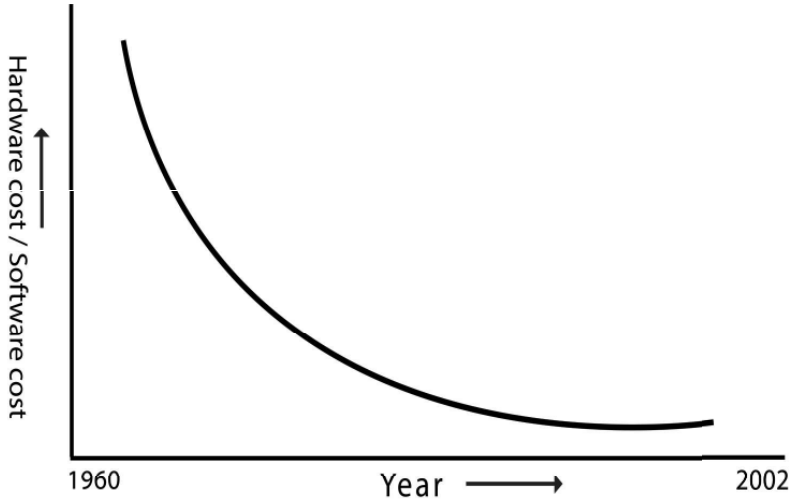
Origin of Software Engineering [1/2]

- Term 'software engineering' was first proposed in 1968 at a NATO conference held to discuss what was then called the software crisis
- It became clear that individual approaches to program development did not scale up to large and complex software systems

Origin of Software Engineering [2/2]

- Software's were unreliable, cost more than expected, and were delivered late.
- Software engineering techniques and methods were developed, such as structured programming, information hiding and object-oriented development.
- Tools and standard notations were developed and are now extensively used.

Software Crisis [1/2]



Software Crisis [2/2]

- 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.

Software Cost Major Problem

- Cost due to ineffective development of the product inefficient resource usage, leading to time and cost over-runs.
- Factors are larger problem sizes, lack of adequate training in software engineering, increasing skill shortage, and low productivity improvements

Essential Attributes of a good Software Application/product

Attributes

- Maintainability
- Dependability and security
- Efficiency
- Acceptability

Unit 1 – Software Engineering

Myths in Software Engineering

Software Myths

- Roger Pressman (1997) describes several common beliefs or myths that software managers, customers, and developers believe falsely.
- He describes these myths as "misleading attitudes that have caused serious problems."
  - Managers Myth
  - Developers Myth
  - Customers Myth

MANAGEMENTS MYTH [1/4]

- MYTH: We already have a book that's full of standards and procedures for building software; won't that provide my people with everything they need to know?
- REALITY: The book of standards may very well exist, but is it used? Are software practitioners aware of its existence? Does it reflect modern software engineering practice? Is it complete? Is it streamlined to improve time to delivery while still maintaining a focus on quality? In many cases, the answer to all of these questions is "no".

MANAGEMENTS MYTH [2/4]

- MYTH: My people have state-of-the-art software development tools; after all, we buy them the newest computers.
- REALITY: It takes much more than the latest model mainframe, workstation, or PC to do high-quality software development. Computer-aided software engineering (CASE) tools are more important than hardware for achieving good quality and productivity, yet the majority of software developers still do not use them effectively

MANAGEMENTS MYTH [3/4]

- MYTH: If we get behind schedule, we can add more programmers and catch up (sometimes called the Mongolian horde concept).
- REALITY: Software development is not a mechanical process like manufacturing, adding people to a late software project makes it later

MANAGEMENTS MYTH [4/4]

- MYTH: If I decide to outsource the software project to a third party, I can just relax and let that firm build it.
- REALITY: If an organization does not understand how to manage and control software projects internally, it will invariably struggle when it outsources software projects.

Introduction to Software Engineering

Developer and Customer Myths

DEVELOPER'S MYTH [1/4]

- MYTH: Once we write the program and get it to work, our job is done.
- REALITY: Someone once said that "the sooner you begin 'writing code', the longer it'll take you to get done. Industry data indicate that between 60 and 80 per cent of all effort expended on software will be expended after it is delivered to the customer for the first time.

DEVELOPER'S MYTH [2/4]

- MYTH: Until I get the program "running", I have no way of assessing its quality.
- REALITY: One of the most effective software quality assurance mechanisms can be applied from the inception of a project—the formal technical review. Software reviews are a "quality filter" that is more effective than testing for finding certain classes of software defects.

DEVELOPER'S MYTH [3/4]

- MYTH: The only deliverable work product for a successful project is the working program.
- REALITY: A working program is only one part of a software configuration that includes many elements. Documentation provides a foundation for successful engineering and, more importantly, guidance for software support.

DEVELOPER'S MYTH [4/4]

- MYTH: Software engineering will make us create voluminous and unnecessary documentation and will invariably slow us down.
- REALITY: Software engineering is not about creating documents. It is about creating quality. Better quality leads to reduced rework. And reduced rework results in faster delivery times.

CUSTOMER'S MYTH [1/2]

- MYTH: A general statement of objectives is sufficient to begin writing programs— we can fill in the details later.
- REALITY: A poor up-front definition is the major cause of failed software efforts. A formal and detailed description of the information domain, function, behavior, performance, interfaces, design constraints, and validation criteria is essential. These characteristics can be determined only after thorough communication between customers and developers.

CUSTOMER'S MYTH [2/2]

- **MYTH: Project requirements continually change, but change can be easily accommodated because the software is flexible.**
- **REALITY: Software requirements indeed change, but the impact of change varies with the time at which it is introduced.**