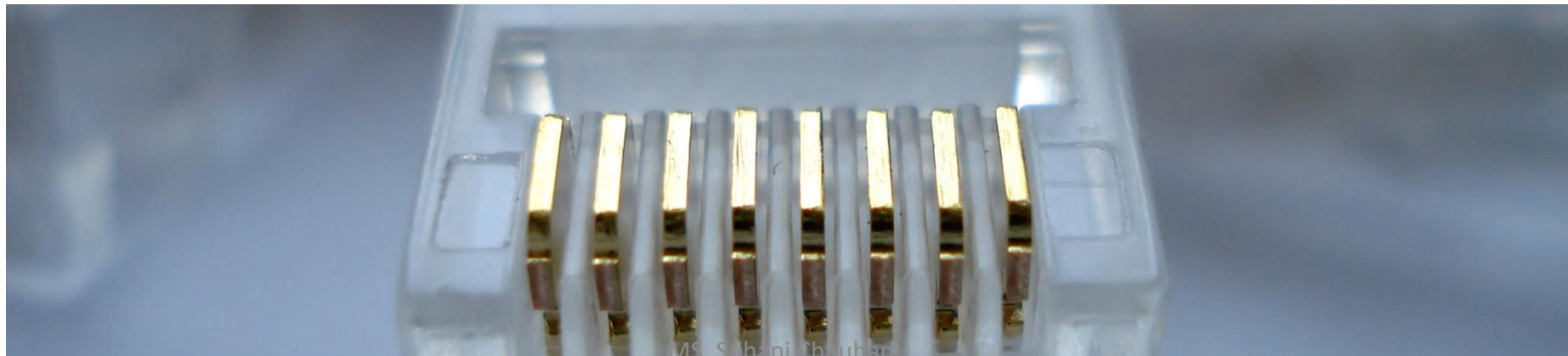


Chapter 1: Introduction to Software Engineering

MS. Suhani Chauhan
SVNIT



Overview

- ◆ Learning Objectives.
- ◆ What is software engineering?
- ◆ Why is software engineering important?

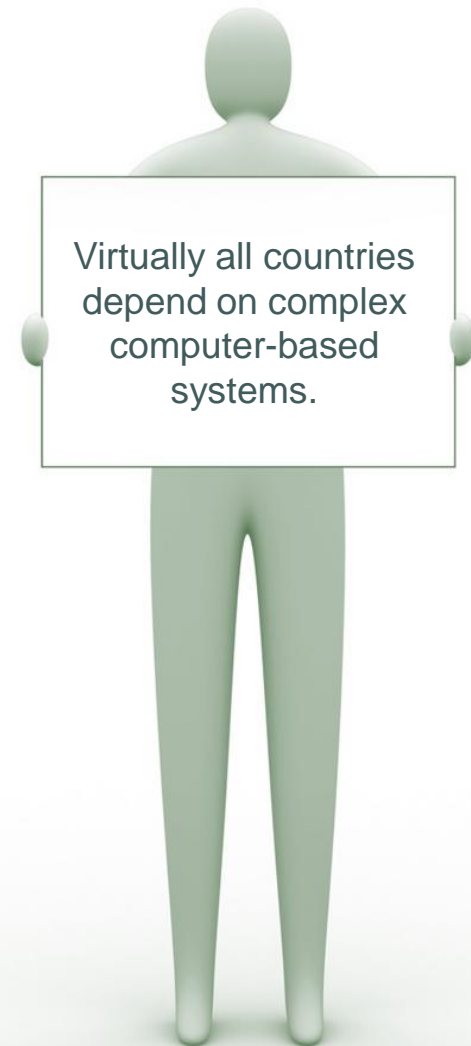
By the end of this chapter, you will...

- ◆ Understand what software engineering is.
- ◆ Understand why software engineering is important.
- ◆ Know answers to key questions related to the software engineering discipline.

Activity

Think about all the devices and systems that you encounter in your everyday life which have software controlling them...

List as many as you can



Software Engineering ?

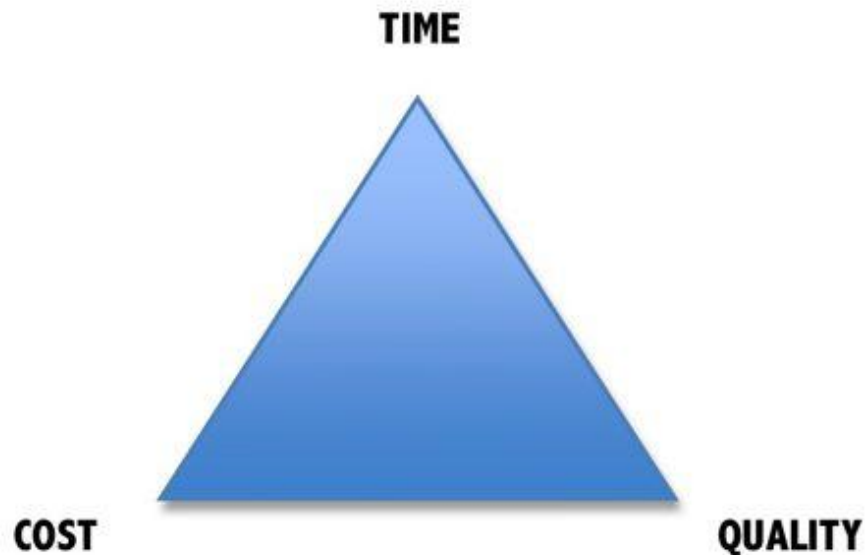
- **Engineering:** Application of science, tools and methods to find cost effective solution to problems.
- **Software engineering:** It is defined as systematic, disciplined and quantifiable approach, for the development, operation and maintenance of software.

Characteristics of S/W

- Software is developed or engineered. It is not manufactured.
- Software does not wear out.
- Software is custom built.

Why is Software Engineering important?

Complex systems need a disciplined approach for designing, developing and managing them.



Software Development Crises

It is a term used in early days of computing science for the difficulty of writing useful & efficient program in required time.

Projects were:

- Over time.
- Over budget.
- Unreliable, low quality.
- Difficult to maintain, inefficient.
- Performed poorly, didn't meet requirement.

Software errors....*the cost*

Errors in computer software can have devastating effects.

Software Crisis

Example 1: 2009, Computer glitch delays flights

Saturday 3rd October 2009-London, England (CNN)

- Dozens of flights from the UK were delayed Saturday after a glitch in an air traffic control system in Scotland, but the problem was fixed a few hours later.
- The agency said it reverted to backup equipment as engineering worked on the system.
- The problem did not create a safety issue but could cause delays in flights.

- Read more at:

<http://edition.cnn.com/2009/WORLD/europe/10/03/uk.flights.delayed>



Software Crisis

Example 2: Ariane 5 Explosion

- European Space Agency spent 10 years and \$7 billion to produce Ariane 5.
- Crash after 36.7 seconds.
- Caused by an overflow error. Trying to store a 64-bit number into a 16-bit space.
- Watch the video:
<http://www.youtube.com/watch?v=z-r9cYp3tTE>



Software Crisis

Example 3: 1992, London Ambulance Service

- Considered the largest ambulance service in the world.
- Overloaded problem.
- It was unable to keep track of the ambulances and their statuses. Sending multiple units to some locations and no units to other locations.
- Generates many exceptions messages.
- 46 deaths.



Therefore...

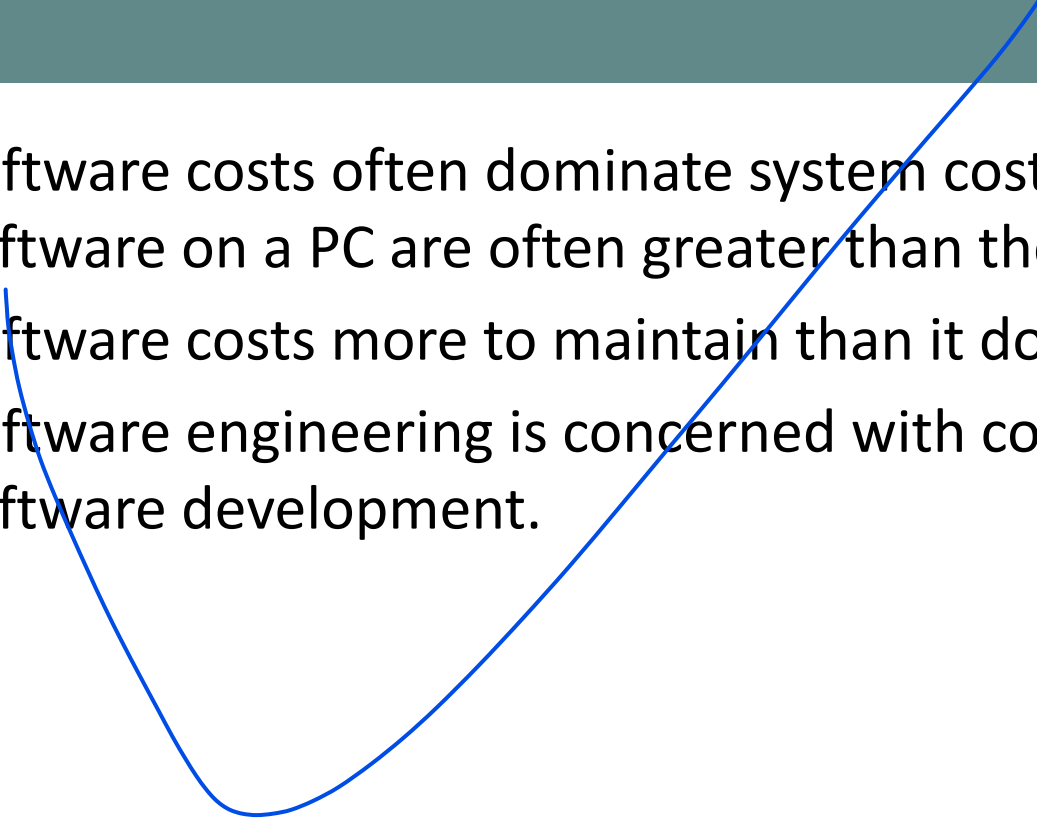
A well-disciplined approach to software development and management is necessary. This is called engineering.

Software Engineering

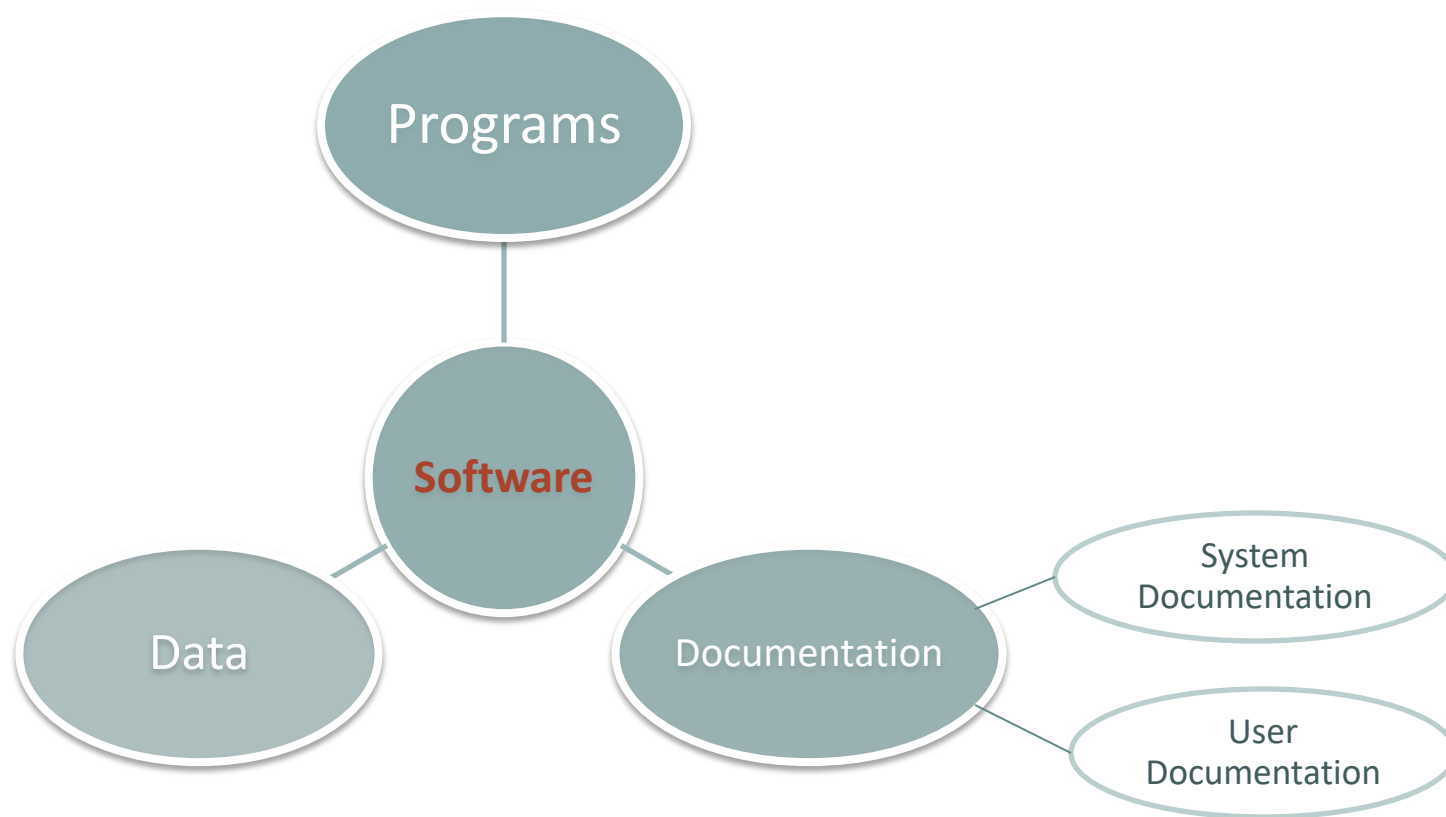
- ♦ The term *software engineering* first appeared in the **1968** NATO Software Engineering Conference and was meant to provoke thought regarding what was then called the “**software crisis**”..
- ♦ “.. An engineering discipline that is concerned **with all aspects of software production** from the **early stages** of system specification to **maintaining the system after it has gone into use.**”

-
- **Software Engineering** is the science and art of building significant software systems that are:
 - 1) on time
 - 2) on budget
 - 3) with acceptable performance
 - 4) with correct operation.
 - The economies of all developed nations are dependent on software.
 - More and more systems are software controlled.
 - Software engineering is concerned with theories, methods and tools for professional software development.

Software Costs

- Software costs often dominate system costs. The costs of software on a PC are often greater than the hardware cost.
 - Software costs more to maintain than it does to develop.
 - Software engineering is concerned with cost-effective software development.
- 

What is Software?



Types of Software

- **Generic products.**
 - Stand-alone systems that are marketed and sold to any customer who wishes to buy them.
 - Examples – PC software such as graphics programs, project management tools; CAD software; software for specific markets such as appointments systems for dentists.
 - The specification of what the software should do is owned by the software developer and decisions on software change are made by the developer.
- **Customized or bespoke products.**
 - Software that is commissioned by a specific customer to meet their own needs.
 - Examples – embedded control systems, air traffic control software, traffic monitoring systems.
 - The specification of what the software should do is owned by the customer for the software and they make decisions on software changes that are required.

Evolution of Software engineering

- 1945-65: Origin, uses build & deliver method.
- 1965-85: Crisis, from 100 only 2 projects were used. Others were in scrap.
- 1990-2000: Internet, famous project MS windows has developed.
- 2000-2010: Mobile OS launched, for light weight application.
- 2010-till: AI,ML,DL came, which are open source and available for free.

Software Engineering vs. Computer Science



“Computer science is no more about computers than astronomy is about telescopes.” *Edsger Dijkstra*

Computer Science

- Theory.
- Fundamentals.

Software Engineering

- Practicalities of software design, development and delivery.

Software Engineering vs. Systems Engineering

Systems Engineering:

- ♦ Interdisciplinary engineering field (computer, software, and process eng.).
- ♦ Focuses on how complex engineering projects should be designed and managed.

Systems Engineering

- All aspects of computer-based systems development: HW + SW + Process.
- Older than SWE.

Software Engineering

- Deals with the design, development and delivery of SW.
- Is part of Systems Engineering.

Frequently asked questions about software engineering

| Question | Answer |
|---|---|
| What is software? | Computer programs and associated documentation. Software products may be developed for a particular customer or may be developed for a general market. |
| What are the attributes of good software? | Good software should deliver the required functionality and performance to the user and should be maintainable, dependable and usable. |
| What is software engineering? | Software engineering is an engineering discipline that is concerned with all aspects of software production. |
| What are the fundamental software engineering activities? | Software specification, software development, software validation and software evolution. |
| What is the difference between software engineering and computer science? | Computer science focuses on theory and fundamentals; software engineering is concerned with the practicalities of developing and delivering useful software. |
| What is the difference between software engineering and system engineering? | System engineering is concerned with all aspects of computer-based systems development including hardware, software and process engineering. Software engineering is part of this more general process. |

Frequently asked questions about software engineering

| Question | Answer |
|--|--|
| What are the key challenges facing software engineering? | Coping with increasing diversity, demands for reduced delivery times and developing trustworthy software. |
| What are the costs of software engineering? | Roughly 60% of software costs are development costs, 40% are testing costs. For custom software, evolution costs often exceed development costs. |
| What are the best software engineering techniques and methods? | While all software projects have to be professionally managed and developed, different techniques are appropriate for different types of system. For example, games should always be developed using a series of prototypes whereas safety critical control systems require a complete and analyzable specification to be developed. You can't, therefore, say that one method is better than another. |
| What differences has the web made to software engineering? | The web has led to the availability of software services and the possibility of developing highly distributed service-based systems. Web-based systems development has led to important advances in programming languages and software reuse. |

What is a Software Process?

♦A software process (also known as software methodology) is a set of related activities that leads to the production of the software. These activities may involve the development of the software from the scratch, or, modifying an existing system.

| SW Process Activity | What is going on there? |
|---------------------|---|
| Specification | What does the customer need? What are the constraints? |
| Development | Design & programming. |
| Validation | Checking whether it meets requirements. |
| Evolution | Modifications (e.g. customer/market). |

What is a Software Process **Model**?

- ◆ Description of the software process that represents one view, such as the activities, data or roles of people involved.

| Examples of views | Focus on... |
|-------------------|---|
| Workflow | Activities = human actions. What is input, output, and dependencies. |
| Dataflow | Activities = transformations of information. How the input is transformed into output. |
| Role/Action | What is the role of people involved in each step of the process? |

Software Process Framework:

- A process framework establishes the foundation for a complete software process by identifying a small number of framework activities that are applicable to all software projects, regardless of size or complexity. It also includes a set of umbrella activities that are applicable across the entire software process.

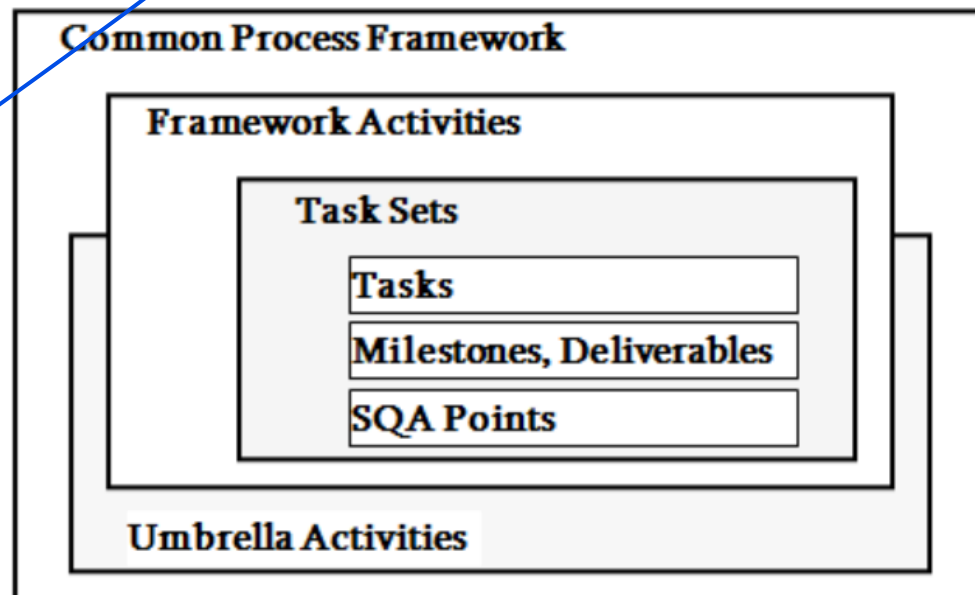


Figure: Chart of Process Framework

Elements of software process.

1. Communication: This activity involves heavy communication with customers and other stakeholders in order to gather requirements and other related activities.
2. Planning: Here a plan to be followed will be created which will describe the technical tasks to be conducted, risks, required resources, work schedule etc.
3. Modeling: A model will be created to better understand the requirements and design to achieve these requirements.
4. Construction: Here the code will be generated and tested.
5. Deployment: Here, a complete or partially complete version of the software is represented to the customers to evaluate and they give feedbacks based on the evaluation.

A PROCESS FRAMEWORK

- Generic view of engineering complimented by a number of **umbrella activities**.
- Software project tracking and control.
- Formal technical reviews.
- Software quality assurance.
- Software configuration management.
- Document preparation and production.
- Reusability management.
- Measurement.
- Risk management

LEGACY SOFTWARE

- Legacy software are older programs that are developed decades ago.
 - inextensible design, convoluted code, poor and nonexistent documentation.
- As time passes legacy systems evolve due to following reasons:
- The software must be
 - adapted to meet the needs of new computing environment or technology.
 - enhanced to implement new business requirements.
 - extended to make it interoperable with more modern systems or database.
 - rearchitected to make it viable within a network environment.

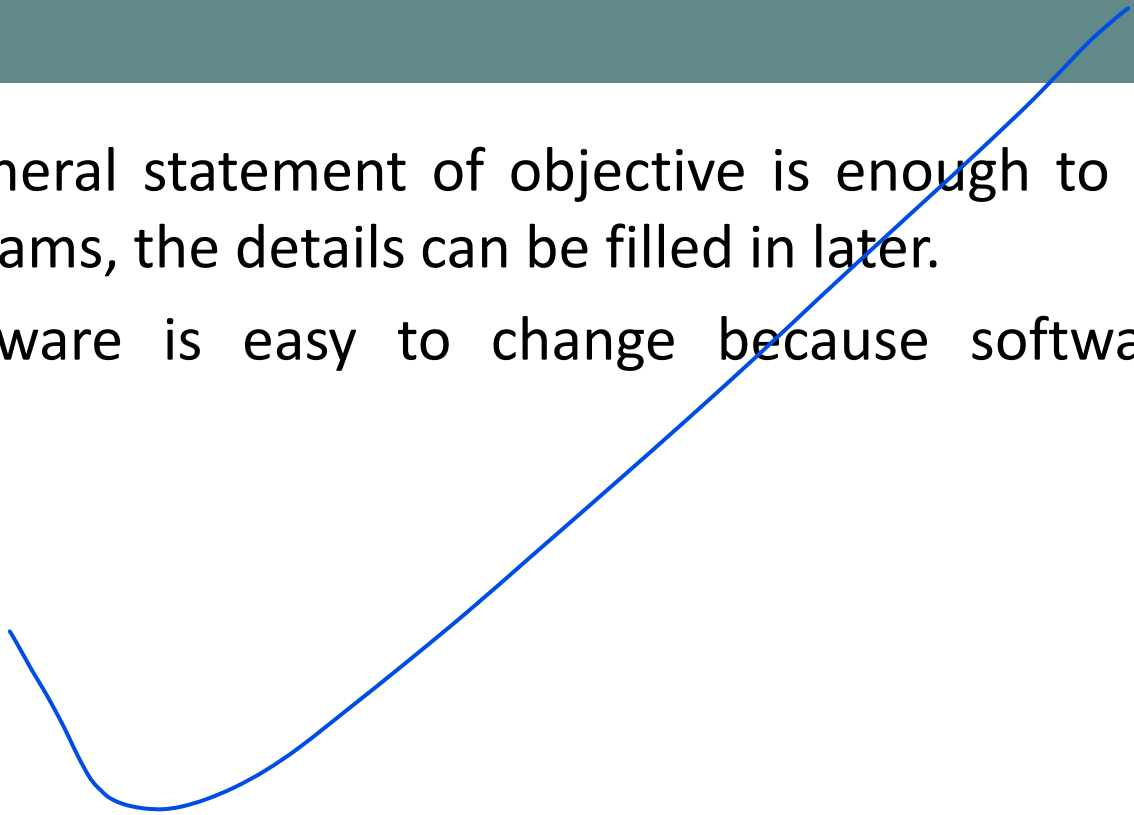
SOFTWARE MYTHS

- Myths are widely held but false beliefs and views which propagate misinformation and confusion.
- Three types of myth are associated with software:
 - Management myth
 - Customer myth
 - Practitioner's myth

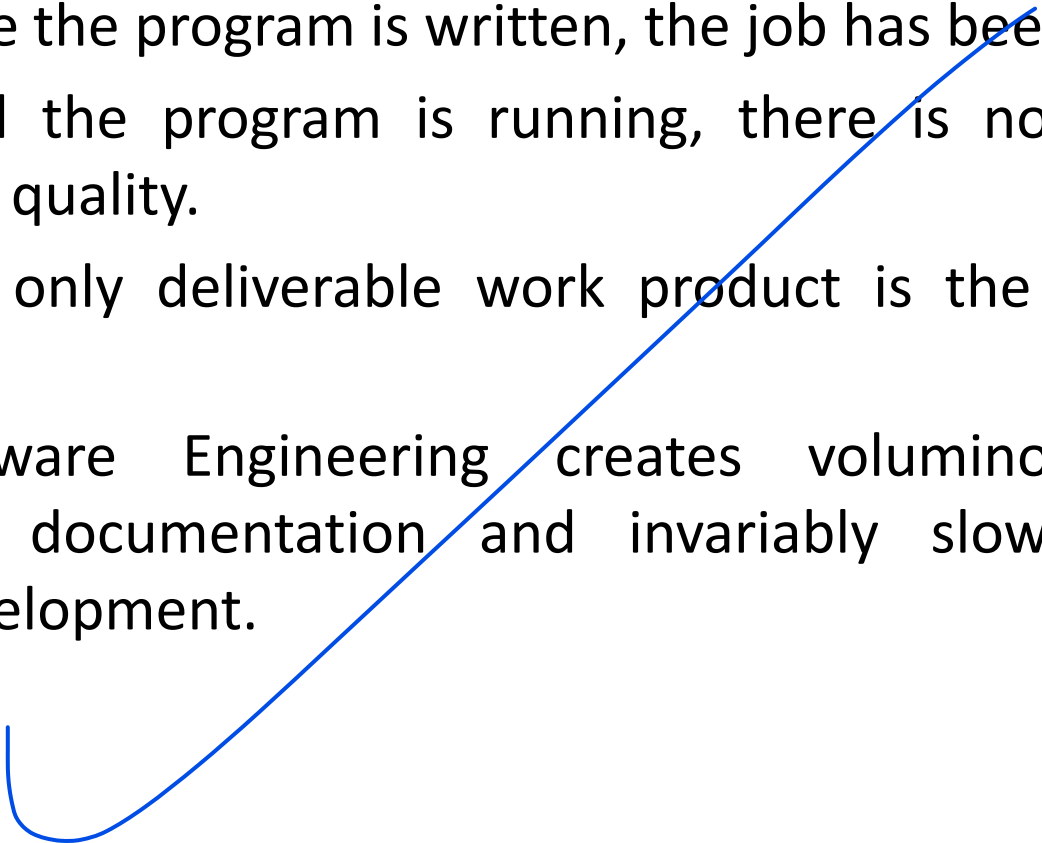
MANAGEMENT MYTHS

- Myth(1)-The available standards and procedures for software are enough.
- Myth(2)-Each organization feel that they have state-of-art software development tools since they have latest computer.
- Myth(3)-Adding more programmers when the work is behind schedule can catch up.
- Myth(4)-Outsourcing the software project to third party, we can relax and let that party build it.

CUSTOMER MYTHS

- Myth(1)- General statement of objective is enough to begin writing programs, the details can be filled in later.
 - Myth(2)-Software is easy to change because software is flexible.
- 

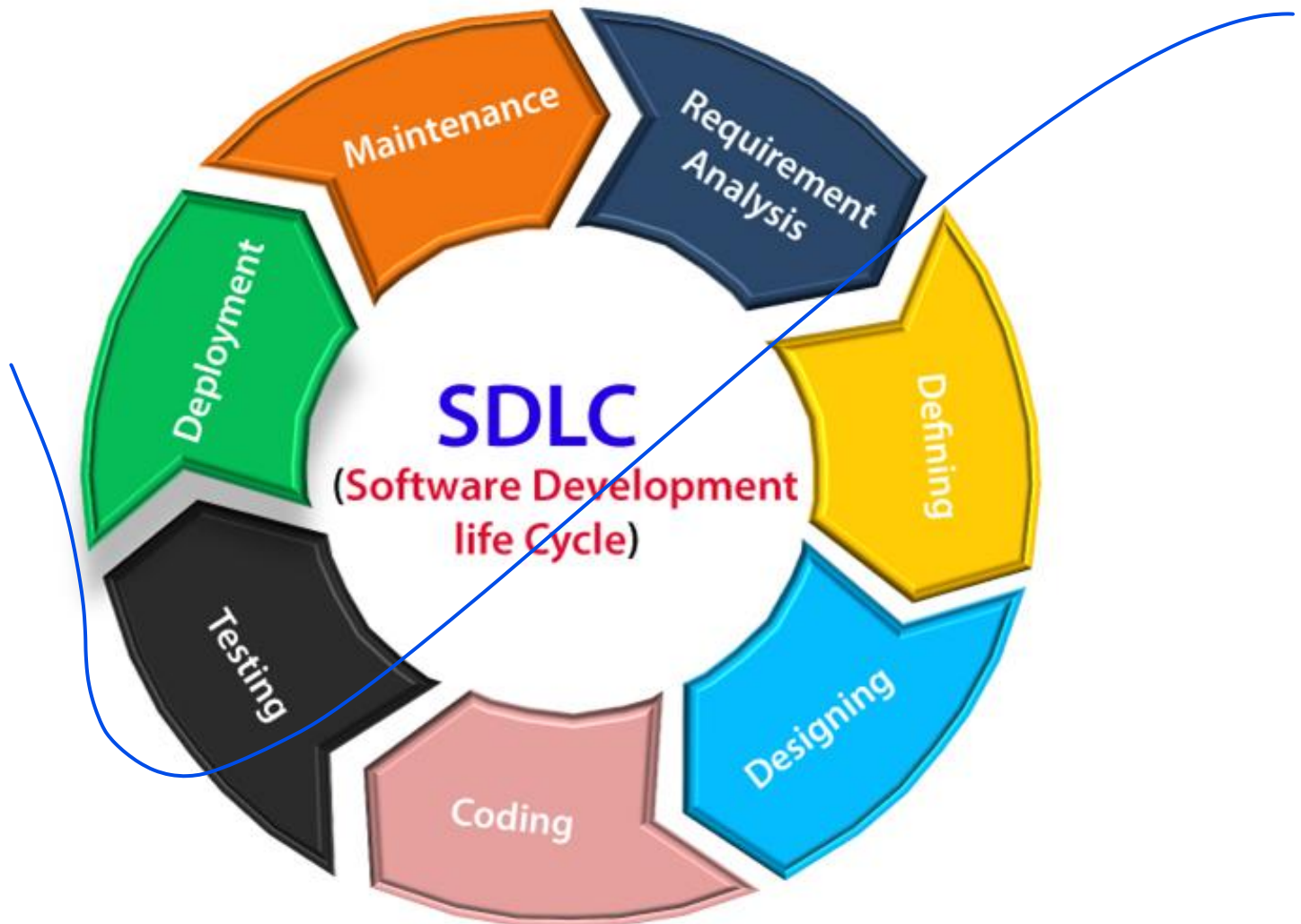
PRACTITIONER'S MYTH

- Myth(1)-Once the program is written, the job has been done.
 - Myth(2)-Until the program is running, there is no way of assessing the quality.
 - Myth(3)-The only deliverable work product is the working program.
 - Myth(4)-Software Engineering creates voluminous and unnecessary documentation and invariably slows down software development.
- 

Software Development Life Cycle.(SDLC)

- Software Development Life Cycle (SDLC) is a process used by the software industry to design, develop and test high quality softwares.
- It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software.
- The life cycle defines a methodology for improving the quality of software and the overall development process.

SDLC



SDLC Stages

- **Requirement analysis:**
 - Communication with user
 - Senior member
 - Identification of risk
 - Ambiguity problem
- **Defining Requirements:**
 - Documenting the product requirement and get approved from customer
 - SRS
- **Designing :**
 - Product architecture
 - DDS(Design document specification)
 - Best of architecture is selected

SDLC Stages

- **Developing :**
 - Code generated
 - Best skilled programming language is selected
 - Use compiler, interpreter, debugger.
- **Testing:**
 - Subset of all stages in SDLC
 - Detecting, reporting, tracking, fixing and retesting
 - Till it reaches SDLC quality standards
- **Deployment:**
 - Releasing the S/W in market.
 - Maintenance is conducted after deployment

Software Quality

- Software quality product is defined in term of its fitness of purpose. That is, a quality product does precisely what the users want it to do.
- For software products, the fitness of use is generally explained in terms of satisfaction of the requirements laid down in the SRS document.
- **Example:** Consider a functionally correct software product. That is, it performs all tasks as specified in the SRS document. But, has an almost unusable user interface. Even though it may be functionally right, we cannot consider it to be a quality product.

Factors of Software Quality

- **Portability:**
 - It can be freely made to work in various OS, in multiple machines, with other software products, etc.
- **Usability:**
 - A software product has better usability if various categories of users can easily invoke the functions of the product.
- **Reusability:**
 - A software product has excellent reusability if different modules of the product can quickly be reused to develop new products.

Factors of Software Quality

- **Correctness:**

- A software product is correct if various requirements as specified in the SRS document have been correctly implemented.

- **Maintainability:**

- A software product is maintainable if bugs can be easily corrected as and when they show up, new tasks can be easily added to the product, and the functionalities of the product can be easily modified, etc.

Attributes of good software

- ◆ Functional attributes (performance; what the system does).
- ◆ Non-functional attributes (quality; how the system does it).

| Product Characteristic | Description |
|------------------------|---|
| Maintainability | Evolution qualities such as Testability, extensibility. |
| Dependability | Reliability, security, safety. |
| Efficiency | Response time, processing time, memory utilization. |
| Usability | Easy to <u>learn</u> how to use the system by target users. <u>Efficient to use</u> the system by users to accomplish a task. <u>Satisfying</u> to use by intended users. |

Challenges facing software engineering

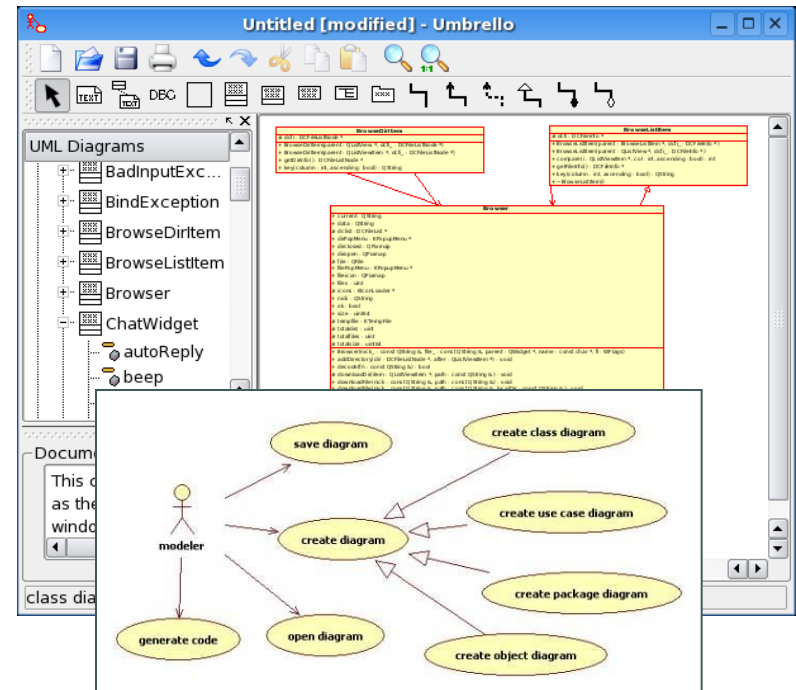
| Challenge | Why? | Software needs to .. |
|----------------------|---|--|
| Heterogeneity | Different computers, different platforms, different support systems. | Cope with this variability. |
| Delivery | Businesses are more responsive → supporting software needs to evolve as rapidly. | Be delivered in shorter time without compromising quality. |
| Trust | Software is a part of many aspects of our lives (work, study, leisure). | Demonstrate that it can be trusted by users. |

What is CASE?

◆ Computer **A**ided **S**oftware **E**ngineering.

◆ Programs that support:

- ◆ Requirements analysis.
- ◆ System modeling.
- ◆ Debugging.
- ◆ Testing.



References

- PRESS&SUN-BULLETIN, The Binghamton Press Co., Binghamton, NY, October 1,1999.
- “Software Hell: Is there a way out?”, BUSINESS WEEK, December 6, 1999.
- *IEEE Standards Collection: Software Engineering, IEEE standard 610.12-1990, IEEE 1993.*
- Sommerville, Ian “*Software Engineering*”, 9th edition, Addison-Wesley.