# Software Engineering

Gunjan Sahni

# Evolving role of software

Today, software takes on a dual role. It is a product and, at the same time, the vehicle for delivering a product. As a product, it delivers the computing potential embodied by computer hardware or, more broadly, a network of computers that are accessible by local hardware

- As the vehicle used to deliver the product, software acts as the basis for the control of the computer (operating systems), the communication of information (networks), and the creation and control of other programs (software tools and environments). Software delivers the most important product of our time—information. Software transforms personal data (e.g., an individual's financial transactions) so that the data can be more useful in a local context;
- it manages business information to enhance competitiveness;
- it provides a gateway to worldwide information networks (e.g., Internet) and provides the means for acquiring information in all of its forms.

# What is SE?

- Software engineering is the systematic approach to the development, operation, maintenance and retirement of software.
- Software Engineering is the application of science and mathematics by which the capabilities of computer equipment are made useful to man via computer programs, procedures, and associated documentations.
- 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

# In Brief

The basic objective of software engineering is to develop methods and procedures for software development that can

- Scale up for large systems
- Can be used consistently to produce high-quality software
- At low cost and
- With a small cycle of time.

# Why Need SE?

- 1. As Software development is expensive so proper measures are required so that the resources are used efficiently and effectively.
- 2. Cost and time considerations are another factor, which arises the need for Software Engineering.
- 3. Reliability factors

# SOFTWARE: A CRISIS ON THE HORIZON

Can be understand by the word crisis, has definition: "the turning point in the course of a disease, when you becomes clear whether the patient will live or die.“

Give clue about the real nature of the problems that have plagued software development

# Software Myths

Software myths propagate false beliefs and confusion in the minds of management, users and developers

# Managers myth

- Managers, who own software development responsibility, are often under strain and pressure to maintain a software budget, time constraints, improved quality, and many other considerations.

| | |
|---|---|
| The members of an organization can acquire all-the information, they require from a manual, which contains standards, procedures, and principles; | •Standards are often incomplete, inadaptable, and outdated.<br>•Developers are often unaware of all the established standards.<br>•Developers rarely follow all the known standards because not all the standards tend to decrease the delivery time of software while maintaining its quality. |
| If the project is behind schedule, increasing the number of programmers can reduce the time gap. | •Adding more manpower to the project, which is already behind schedule, further delays the project.<br>•New workers take longer to learn about the project as compared to those already working on the project. |
| If the project is outsourced to a third party, the management can relax and let the other firm develop software for them. | •Outsourcing software to a third party does not help the organization, which is incompetent in managing and controlling the software project internally. The organization invariably suffers when it out sources the software project. |

# Users Myths

Users tend to believe myths about the software because software managers and developers do not try to correct the false beliefs. These myths lead to false expectations and ultimately develop dissatisfaction among the users.

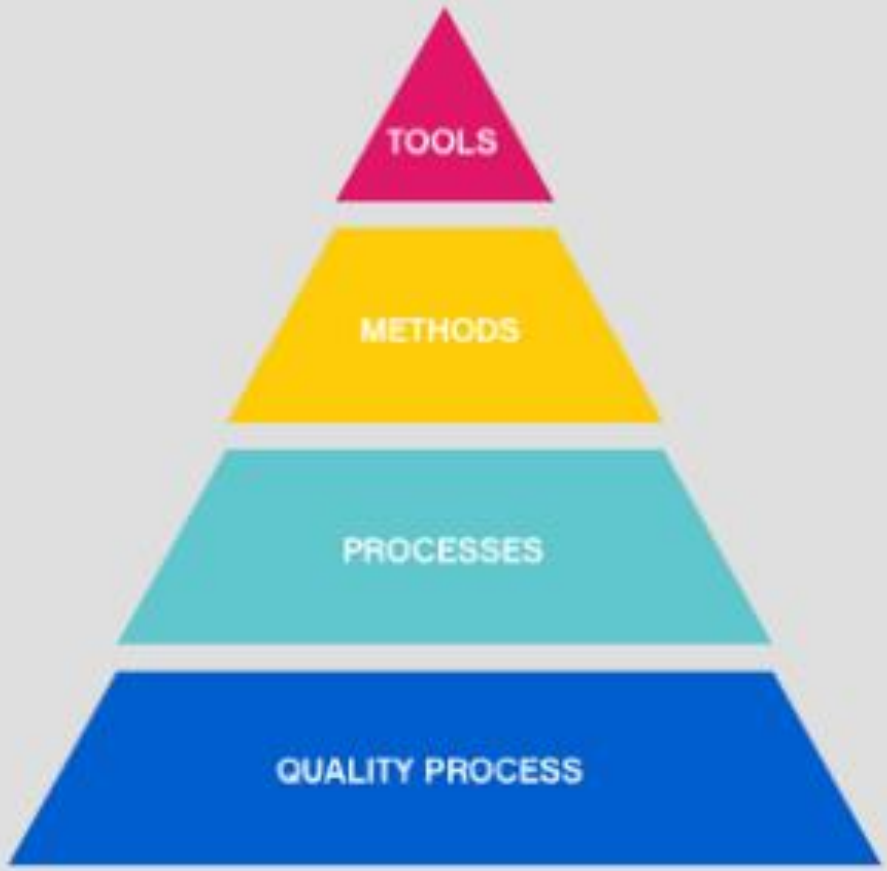
| | |
|---|---|
| •Brief requirement stated in the initial process is enough to start development; detailed requirements can be added at the later stages. | •Starting development with incomplete and ambiguous requirements often lead to software failure. Instead, a complete and formal description of requirements is essential before starting development.<br>•Adding requirements at a later stage often requires repeating the entire development process. |
| •Software is flexible; hence software requirement changes can be added during any phase of the development process. | •Incorporating change requests earlier in the development process costs lesser than those that occurs at later stages. This is because incorporating changes later may require redesigning and extra resources. |

# Developer Myths

| | |
|---|---|
| •Software development is considered complete when the code is delivered. | •50% to 70% of all the efforts are expended after the software is delivered to the user. |
| •The success of a software project depends on the quality of the product produced. | •The quality of programs is not the only factor that makes the project successful instead the documentation and software configuration also playa crucial role. |
| •Software engineering requires unnecessary documentation, which slows down the project. | •Software engineering is about creating quality at every level of the software project. Proper documentation enhances quality which results in reducing the amount of rework. |
| •The only product that is delivered after the completion of a project is the working program(s). | •The deliverables of a successful project includes not only the working program but also the documentation to guide the users for using the software. |
| •Software quality can be assessed only after the program is executed. | •The quality of software can be measured during any phase of development process by applying some quality assurance mechanism. One such mechanism is formal technical review that can be effectively used during each phase of development to uncover certain errors. |

# SOFTWARE ENGINEERING

## is a layered technology



Software engineering can be viewed as a layered technology. Various layers are listed below:

- The **process layer** allows the development of software on time. It defines an outline for a set of key process areas that must be acclaimed for effective delivery of software engineering technology.
- The **method layer** provides technical knowledge for developing software. This layer covers a broad array of tasks that include requirements analysis, design, coding, testing, and maintenance phase of the software development.
- The **tools layer** provides computerized or semi-computerized support for the process and the method layer. Sometimes tools are integrated in such a way that other tools can use information created by one tool.
- This **Quality Focus** layer is the fundamental layer for software engineering. As stated above it is of great importance to test the end product to see if it meets its specifications. Efficiency, usability, maintenance and reusability are some of the requirements that need to be met by new software