## INTRODUCTION TO SOFTWARE ENGINEERING

### Software Engineering (SE)

- × The term software engineering was popularized during the 1968 NATO Software Engineering Conference (held in Garmisch, Germany) by its chairman F.L. Bauer, and has been in widespread use since.
- × The field of SE aims to find answers to the many problems that software development projects are likely to meet when constructing large software systems.
- × Such systems are complex
  - because of their sheer size,
  - because they are developed by a team involving people from different disciplines, and
  - because they will be modified regularly to meet changing requirements, both during development and after installation.
- × The software developed is according to:
  - → Accepted industry practice, with good quality control, adherence to standards, and in an efficient and timely manner.
- × The term **software engineering** refers to a systematic procedure that is used in the context of a generally accepted set of goals for the analysis, design, implementation, testing and maintenance of software.
- × According to IEEE **SOFTWARE ENGINEERING** is the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software.
- × The discipline of software engineering encompasses knowledge, tools, and methods for defining software requirements, and performing software design, software construction, software testing, and software maintenance tasks.
- × **SOFTWARE ENGINEERING** also draws on knowledge from fields such as computer engineering, computer science, management, mathematics, project management, quality management, software ergonomics, and systems engineering. This is according to the author Ronald Leach.
- × The software produced should be efficient, reliable, usable, modifiable, portable, testable, reusable, maintainable, interoperable, and correct.
- × SWEBOOK (Software Engineering Body of Knowledge), 2013 defines these characteristics as the following
- × The software produced should be:
  - **Efficient** – software made in expected time and within resources
  - **Reliable** – software performs as expected
  - **Usable** – software can be used properly
  - **Modifiable** – software can be easily changed as requirement changes
  - **Portable** – system can be ported to other computers without major rewriting of the software
  - **Testable** – software can be easily tested
  - **Reusable** – software can be used again in other projects
  - **Maintainable** – software can be easily understood and changed overtime if problems occur
  - **Interoperable** – software can interact with other systems properly
  - **Correct** – the program produces the correct output

**Where Does The Software Engineer Fit In?**

**Software Engineering:**

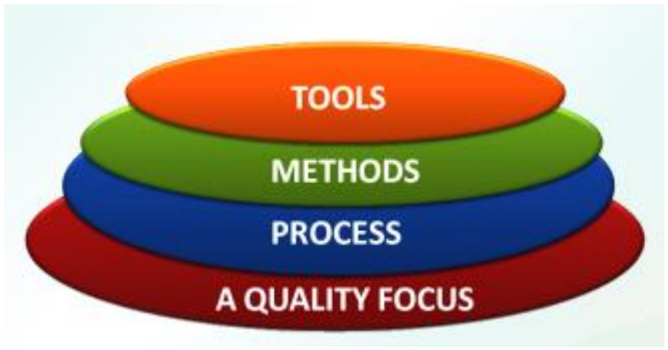
- focusing on computer as a problem-solving tool
- Relationship between computer science and software engineering.

**But the question is... why is there a need for software engineering?**

Over the past years

- Technology has advanced rapidly
  - Ordering of food, clothes, or even transportation is done through smartphones, while on the phone and en route to your next destination
- With this example, software has grown to provide humanity with endless opportunities
- More complex is the system, the more is the programming skill needed.
- In a nutshell, software engineering is developing complex software end products by professionals within time and budget.
- Software engineering is often viewed as layered.

**SOFTWARE ENGINEERING IS A LAYERED TECHNOLOGY AS FOLLOWS:**



## SOFTWARE ENGINEERING LAYERS

### Quality Focus

→ The bedrock that supports software engineering. Any engineering approach must rest on an organizational commitment to quality.

### Process

→ It is the glue that holds the technology layers together and enables rational and timely developments of computer SW.

→ Process defines a framework that must be established for effective delivery of SW engineering technology.

→ It forms the basis for
- management control of software projects
- establishes the context (technical methods applied)
- Work products (models, documents, data, reports, form, etc.) are produced,
- milestones are established,
- quality is ensured, and change is properly managed

### Methods

→ Provide the technical how-to's for building SW.

→ Encompasses a broad array of tasks that include communication, requirements analysis, design modeling, program construction, testing, and support.

→ Rely on a set of basic principles that govern each area of the technology and include modeling activities and other descriptive techniques.

### Tools

→ Provide automated and semi automated support for the process and the methods.

→ When tools are integrated so that information created by one tools can be used by another, a system for the support of SW development, called CASE (Computer-Aided Software Engineering), is established.

## CHALLENGES IN DEVELOPING SOFTWARE

1. Developing / large / complex software application
2. Effort intensive
3. High cost (in case the solution is unsuccessful)
4. Long development time
5. Changing needs for users
6. High risk of failure, reasons of user acceptance, performance and maintainability

**What is Good Software? Perspective on Quality**

Dr. David A. Garvin, PhD, a professor of Business Administration Harvard Business school identified 5 major approaches of defining quality

- The **transcendent** view
- The **product-based** view
- The **user-based** view
- The **manufacturing-based** view
- The **value-based** view

## QUALITY: Transcendent

**Transcendent**

› Quality is an innate excellence (cannot be defined exactly
› Quality is simple, unanalyzable recognized only through experience

**Product-Based**

› Quality is precise, measurable variable in the components and attributes of a product
› Reflects the presence or absence of such measurable and desired product attributes

## QUALITY: User-Based

**User-based**

› Goods that best satisfies user preferences have the highest quality

## QUALITY: Manufacturing-Based

**Manufacturing based**

› Conformance to requirements
› Preventing defects (less expensive than repair/rework)
› Products that follows specification are of high quality

## QUALITY: Value-based

**Value Based**

› Provides quality product at an acceptable price or conformance at an acceptable cost
› This approach is becoming the more prevalent

## What is Good Software?

× Good software engineering must always include a strategy for producing quality software.
× Three ways of considering quality
  - The quality of the product
  - The quality of the process
  - The quality of the product in the context of the business environment

## The Quality of the Product

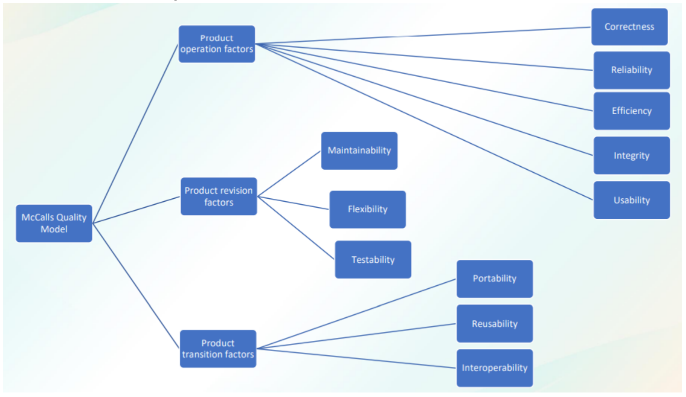
✓ Users judge external characteristics (e.g., correct functionality, number of failures, type of failures)
✓ Designers and maintainers judge internal characteristics (e.g., types of faults)
✓ Different stakeholders may have different criteria
✓ Need quality models to relate the user's external view to developer's internal view
✓ Classic model of software quality factors was suggested by McCall (McCall et al., 1977)
✓ McCall's factors consists of 11 software quality factors

- ✓ Similarly models suggested by Deutsch and Willis (1988) and Evans and Marciniak (1987)
- ✓ All these models do not differ substantially from McCall's model
- ✓ McCall factor model provides a practical, up-to-date method for classifying software requirements (Pressman, 2000).

### McCall's Quality Model

1. Correctness
2. Reliability
3. Efficiency
4. Integrity
5. Usability
6. Maintainability
7. Flexibility
8. Testability
9. Portability
10. Reusability
11. Interoperability

- ✓ The model was grouped into three categories (product operation factors, product revision factors, product transition factors)



- ✓ These qualities are also subdivided based on the external and internal qualities
- ✓ External quality is the quality of the finished product, the quality as it appears to the external world, as it comes of the end of the assembly line. These are factors that are clearly visible to end users.
- ✓ Internal quality is the quality of the product as it is being constructed, while it is on the assembly line. It is concerned with internal technical issues of the software.
- ✓ In general, users of the software only care about external qualities, but it is the internal qualities – which deal largely with the structure of the software – that help developers achieve external qualities.

| | |
|---|---|
| External Quality | Integrity<br>Reliability<br>Usability<br>Correctness |
| Internal Quality | Efficiency<br>Maintainability<br>Testability<br>Flexibility<br>Interoperability<br>Reusability<br>Portability |

## The Quality of the Process

- ✓ Quality of the development and maintenance process is as important as the product quality
- ✓ The development process needs to be modeled
- ✓ Modeling will address questions such as
  - › Where to find a particular kind of fault
  - › How to find faults earlier
  - › How to build in fault tolerance
  - › What are alternative activities
- ✓ This is similar to the process called software quality management (SQM)
- ✓ SQM is a management process that aims to develop and manage the quality of the software in such a way so to best ensure that the product meets the quality standards expected by the customer while also meeting any necessary regulatory and developer requirements is any

- ✓ The following are the Quality management activities that can ensure the quality of the software
  1. Quality assurance
  2. Quality planning
  3. Quality control
- ✓ The following are the Quality management activities that can ensure the quality of the software
  - × **Quality assurance**
    - sets up an organized and logical set of organizational processes and deciding on that software development standards
    - industry best practices paired with those organizational processes, software developers stand a better chance of producing higher quality software.
  - × **Quality planning**
    - Quality planning works at a more granular, project-based level,
    - defining the quality attributes to be associated with the output of the project, and
    - how those attributes should be assessed.
  - × **Quality control**
    - The quality control team tests and reviews software at its various stages to ensure quality assurance processes and standards at both the organizational and project level are being followed.

## The Quality in the Context of the Business Environment

- ✓ Models for process improvement
  - › SEI's Capability Maturity Model (CMM)
  - › ISO 9000
  - › Software Process Improvement and Capability dEtermination (SPICE)
- ✓ Business value is as important as technical value
- ✓ Business value (in relationship to technical value) must be quantified
- ✓ A common approach: **return on investment (ROI)**
  - › what is given up for other purposes
- ✓ ROI is interpreted in different terms: reducing costs, predicting savings, improving productivity, and costs (efforts and resources)

## REASONS FOR SOFTWARE FAILURE

- schedule slippage (software is not really time)
- cost overruns (the software was abandoned in the middle because of the high costing requirement)
- does not solve user's problem (not useful)
- poor quality
- poor maintainability

## What Leads To Such Problems

- No planning of development work
- Deliverables to user not identified
- Poor understanding of user requirement
- No control of review (review and control systematically; cost time and resources)
- Technical incompetence (the need to understand the changes in technologies, new tools and techniques in recent trends in technologies )
- Poor understanding of cost and effort both from user and the developer.

## CHALLENGES FACING SOFTWARE ENGINEERING

- Legacy challenge.
- Heterogeneity challenge.
- Delivery challenge.

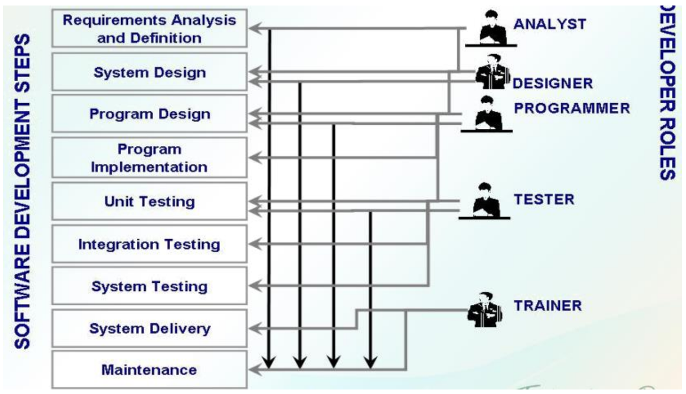
## Who Does Software Engineering?

- **Customer**: the company, organization, or person who pays for the software system
- **Developer**: the company, organization, or person who is building the software system
- **User**: the person or people who will actually use the system

## MEMBERS OF THE DEVELOPMENT TEAM

- **Requirement Analysts** – work with the customers to identify and document the requirements
- **Designers** - generate a system-level description of what the system is suppose to do
- **Programmers** - write lines of code to implement the design
- **Testers** - catch faults
- **Trainers** - show users how to use the system
- **Maintenance Team** - fix faults that show up later
- **Librarians** - prepare and store documents such as software requirements
- **Configuration Management Team** - maintain correspondence among various artifacts

- Typical roles played by the members of a development team


SOFTWARE DEVELOPMENT STEPS
Requirements Analysis and Definition
System Design
Program Design
Program Implementation
Unit Testing
Integration Testing
System Testing
System Delivery
Maintenance
ANALYST
DESIGNER
PROGRAMMER
TESTER
TRAINER
DEVELOPER ROLES