Definitions

**Software** is the product that software professionals build and then support over the long term. Software encompasses (a) instructions, function, and performance, (b) data structures that enable the programs to adequately store and manipulate information and (c) documentation that describes the operation and use of the programs.

**Software engineering** is concerned with theories, methods and tools for professional software development. It is the science and art of building significant software systems that are on time, on budget, with correct operations and with acceptable performance. Software costs often dominate system costs; software costs more to maintain than it does to develop. Software engineering forms a bridge from customer needs to programming implementation. Software engineering is concerned with cost-effective software development.

‘Software Engineering: (1) The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software. (2) The study of approaches as in (1).’ IEEE definition.
Stakeholders
A **customer** requires a system to achieve some business goals by user interaction or interaction with the environment in a specific manner. **Software engineers** have to (1) understand how the system-to-be needs to interact with the user or the environment so that customer requirements are met and (2) design the software-to-be. The **programmer** has to implement the software-to-be designed by the software engineer.

Software products
1. 1:1 developed products: systems which are commissioned by a specific customer and developed specially by some contractor.
2. 1:N developed (generic) products: Stand-alone systems which are produced by a development organization and sold on the open market to any customer.
3. M:N (hybrid) systems: generic products sold as stand alone, customized according to customer needs.

Software product evaluation criteria
Maintainability, dependability, effectiveness, efficiency, usability, hybridness.

The relative importance of these evaluation criteria (attributes of software products) depends on the product and the environment in which it is to be used. In some cases, some attributes may dominate.

Software engineering focuses on a layered technology, based on the layers (a) quality focus, (b) process, (c) methods and (d) tools.

Process layer as the foundation defines a framework with activities for effective delivery of software engineering technology. Method provides technical how-to's for building software. Tools provide automated or semi-automated support for the process and methods. The quality focus is the general commitment of an organization to quality which fosters a continuous process improvement culture.
Attributes of good software

<table>
<thead>
<tr>
<th>Maintainability</th>
<th>Software should be written in such a way so that it can evolve to meet the changing needs of customers. This is a critical attribute because software change is an inevitable requirement of a changing business environment.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Efficiency</td>
<td>Software should not make wasteful use of system resources such as memory and processor cycles. Efficiency therefore includes responsiveness, processing time, memory utilization, etc.</td>
</tr>
<tr>
<td>Acceptability</td>
<td>Software must be acceptable to the type of users for which it is designed. This means that it must be understandable, usable and compatible with other systems that they use.</td>
</tr>
<tr>
<td>Dependability and security</td>
<td>Software dependability includes a range of characteristics including reliability, security and safety. Dependable software should not cause physical or economic damage in the event of system failure. Malicious users should not be able to access or damage the system.</td>
</tr>
</tbody>
</table>

Software applications

- System software: compilers, editors, file management utilities
- Application software: stand-alone programs for specific needs
- Engineering software: Characterized by ‘number crunching’ algorithms
- Embedded software: resides within a product or system
- Product-line software: focus on a limited marketplace to address mass consumer market.
- Web applications: network centric software.
- Artificial Intelligence (AI) software: uses non-numerical algorithm to solve complex problem.

Software development process

Structured set of activities required to develop a software system (specification, design, validation, evolution), which vary depending on the organization and the type of system being developed.

Software development methods

- waterfall
- iterative
Classification of software process models

Generic software process models
Waterfall: Separate and distinct phases of specification and development.
Evolutionary: Specification and development are interleaved.
Formal Transformation: A mathematical system model is formally transformed to an implementation.
Reuse-based: The system is assembled from existing components.

Engineering process model
Specification: Set out the requirements and constraints on the system.
Design: Produce a model of the system.
Manufacture: Build the system.
Test: Check the system meets the required specifications.
Install: Deliver the system to the customer and ensure it is operational.
Maintain: Repair faults in the system as they are discovered.

Hybrid process models
Large systems are usually made up of several sub-systems. The same process model need not be used for all subsystems. Prototyping for high-risk specifications. Waterfall model for well-understood developments.

Spiral model
Focuses attention on reuse options. Focuses attention on early error elimination.
Puts quality objectives up front. Integrates development and maintenance.
Provides a framework for hardware/software development. Contractual development often specifies process model and deliverables in advance.
Requires risk assessment expertise.
Potential problems of process models

<table>
<thead>
<tr>
<th>Model</th>
<th>Visibility</th>
<th>Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Waterfall</td>
<td>High risk for new systems because of specification and design problems. Low risk for well-understood developments using familiar technology. Unidirectional, no way back, finish this step before moving to the next</td>
<td></td>
</tr>
<tr>
<td>Prototyping</td>
<td>Low risk for new applications because specification and program stay in step. High risk because of lack of process visibility.</td>
<td></td>
</tr>
<tr>
<td>Transformational</td>
<td>High risk because of need for advanced technology and staff skills.</td>
<td></td>
</tr>
</tbody>
</table>

Process visibility

Software systems are intangible so managers need documents to assess progress.

Waterfall model is still the most widely used model.

<table>
<thead>
<tr>
<th>Model</th>
<th>Visibility</th>
<th>Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>waterfall</td>
<td>good</td>
<td>Each activity produces some deliverable</td>
</tr>
<tr>
<td>Evolutionary development</td>
<td>poor</td>
<td>Uneconomic to produce documents during rapid iteration</td>
</tr>
<tr>
<td>Formal transformations</td>
<td>good</td>
<td>Documents must be produced for each phase of the process to continue</td>
</tr>
<tr>
<td>Reuse-oriented development</td>
<td>moderate</td>
<td>Sometimes artificial to produce documents describing reuse and reusable components</td>
</tr>
<tr>
<td>Spiral model</td>
<td>good</td>
<td>Each segment of the spiral should produce some document</td>
</tr>
</tbody>
</table>

Unified modeling language (UML) to understand the problem domain

- System to be developed
- Actors
- Agents external to the system
• Concepts/Objects
• Agents working inside the system
• Use Cases
• Scenarios for using the system

**Software measurement**

What to measure?

• Project (developer’s work), for budgeting and scheduling
• Product, for quality assessment

Software engineering in practice

1. Understand the problem (communication and analysis):
   - Who has a stake in the solution to the problem?
   - What are the unknowns?
   - Can the problem be compartmentalized?

2. Can the problem be represented graphically?
   - Plan a solution (modeling and software design)
   - Have you seen similar problems before?
   - Has a similar problem been solved?
   - Can sub problems be defined?
   - Can you represent a solution in a manner that leads to effective implementation?

3. Carry out the plan (code generation)
   - Does the solutions conform to the plan?
   - Is each component part of the solution provably correct?

4. Examine the result for accuracy (testing and quality assurance)
   - Is it possible to test each component part of the solution?
   - Does the solution produce results that conform to the data, functions, and features that are required?
### Questions on Software engineering

<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>What is software?</td>
<td>Computer programs, data structures and associated documentation. Software products may be developed for a particular customer or may be developed for a general market.</td>
</tr>
<tr>
<td>What are the attributes of good software?</td>
<td>Good software should deliver the required functionality and performance to the user and should be maintainable, dependable and usable.</td>
</tr>
<tr>
<td>What is software engineering?</td>
<td>Software engineering is an engineering discipline that is concerned with all aspects of software production.</td>
</tr>
<tr>
<td>What is the difference between software engineering and system engineer-</td>
<td>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.</td>
</tr>
<tr>
<td>ing?</td>
<td></td>
</tr>
<tr>
<td>What is the difference between software engineering and computer science?</td>
<td>Computer science focuses on theory and fundamentals; software engineering is concerned with the practicalities of developing and delivering useful software.</td>
</tr>
</tbody>
</table>