Software Engineering (20CS440)

The Presentation Slides are Influenced by the Text Book *Software Engineering: A Practitioner's Approach, 8/e* (McGraw-Hill)

Dr. Trisiladevi C. Nagavi

Associate Professor

Unit I : Software Process

(Software and Software Engineering)

Chapter 1:The Nature of Software

Chapter 2: Software Engineering

Chapter 3: Software Process Structure

Chapter 4: Process Models

Chapter 2: Software Engineering

- **2.1 Defining the Discipline**
- **2.2 The Software Process**
  - 2.2.1 The Process Framework
  - 2.2.2 Umbrella Activities
  - 2.2.3 Process Adoption
- **2.3 Software Engineering Practice**
  - 2.3.1 The Essence of Practice
  - 2.3.2 General Principles

2.1 Defining the Discipline

The seminal definition:

[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.

The IEEE definition:

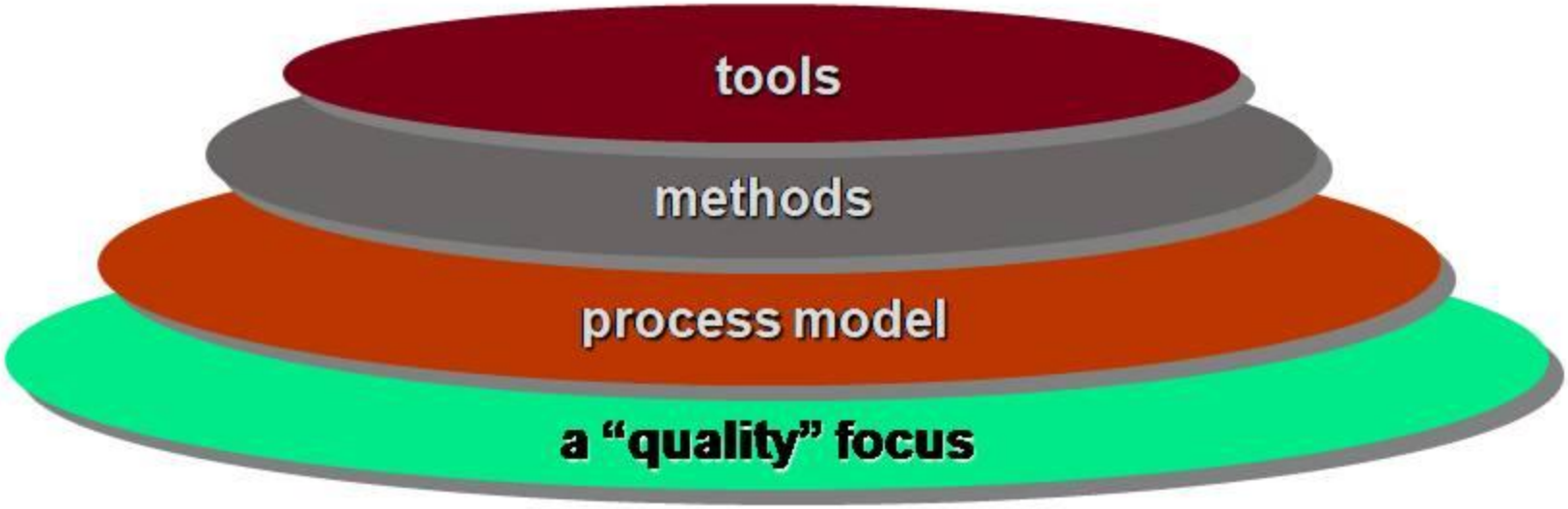
Software Engineering:

(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).

2.1 Defining the Discipline

Foundation for software Engineering is a **layered technology**



Software Engineering

2.1 Defining the Discipline

Various tasks required to build software – **e.g. design, testing**

- **SE process:** Organization & management of tasks: **framework**
- Forms the basis for **management control, technical methods, output (models, documents, data, reports, forms) produced, milestones established, quality ensured, change managed**.

- **SE methods:** ways to perform tasks – e.g. methods for testing, design etc. It indicate technical how to's
- **SE tools:** assist in performing the tasks – e.g. design tools, IDEs like Eclipse – UML tools: Rational Rose, – Testing tools like JUnit, Jcover (CASE). Automated or semi –automated tools.

2.2 Software Process

- A process is a collection of activities, actions and tasks that are performed when some work product is to be created. It is **not a rigid prescription** for how to build computer software. Rather, it is an **adaptable** approach that enables the people doing the work to pick and choose the **appropriate set of work actions** and tasks.
- Activity communication with customer, action is design and task is unit testing as the examples.

- Purpose of process is to deliver software in a timely manner and with sufficient quality to satisfy those who have sponsored its creation and those who will use it.

2.2.1 The Process Framework

Process framework
- **Framework activities**
  - work tasks
  - work products
  - milestones & deliverables
  - QA checkpoints
- **Umbrella Activities**

The Process Framework establishes the foundation for complete SE process with **framework activities**. Also encompasses **umbrella activities** that are applicable to entire software process.

2.2 Software Process

2.2.1 The Process Framework: 5 Activities of a Process framework

- **Communication:** with customer to understand objectives and gather requirements.
- **Planning:** creates a "map" describing the tasks, risks and resources, work products and work schedule.
- **Modeling:** Create a "sketch", what it looks like architecturally, how the constituent parts fit together and other characteristics.
- **Construction:** code generation and the testing.
- **Deployment:** Delivered to the customer, he evaluates the products and provides feedback.

- Activities used regardless of the **application domain, size of the project, complexity of the efforts** etc.
- Framework activities are applied **iteratively.**

2.2 Software Process

2.2.2 Umbrella Activities

Complement the five process framework activities and help team manage and control progress, quality, change, and risk.

1. Software project tracking and control: assess progress against the plan and take actions to maintain the schedule.
2. Risk management: assesses risks that may affect the outcome and quality.

3. Software quality assurance: defines and conduct activities to ensure quality.
4. Technical reviews: assesses work products to uncover and remove errors before going to the next activity.

5. Measurement: define and collects process, project, and product measures to ensure stakeholder's needs are met.
6. Software configuration management: manage the effects of change throughout the software process.

7. Reusability management: defines criteria for work product reuse and establishes mechanism to achieve reusable components.
8. Work product preparation and production: create work products such as models, documents, logs, forms and lists.

2.2 Software Process

2.2.3 Process Adoption : Every Project is different in terms of

1. the **overall flow** of activities, actions, and tasks and the interdependencies among them
2. the **degree to which actions and tasks are defined** within each framework activity
3. the degree to which work **products are identified and required**
4. the manner which **quality assurance activities** are applied
5. the manner in which **project tracking and control** activities are applied
6. the overall degree of **detail and rigor** with which the process is described
7. the degree to which the customer and other **stakeholders are involved** with the project
8. the level of **autonomy** given to the software team
9. the degree to which **team organization** and roles are prescribed

2.3 Software Engineering Practice

2.3.1 The Essence of Practice

- **Relation** between practice of software engineering (communication, planning, modeling, construction and deployment) and **problem solving.**

- George Polya outlines the essence of problem solving, :
  1. Understand the problem (communication and analysis).
  2. Plan a solution (modeling and software design).
  3. Carry out the plan (code generation).
  4. Examine the result for accuracy (testing and quality assurance).

1. Understand the Problem

- *Who has a stake in the solution to the problem?* That is, who are the stakeholders?
- *What are the unknowns?* What data, functions, and features are required to properly solve the problem?
- *Can the problem be compartmentalized?* Is it possible to represent smaller problems that may be easier to understand?
- *Can the problem be represented graphically?* Can an analysis model be created?

2. Plan the Solution

- *Have you seen similar problems before?* Are there patterns that are recognizable in a potential solution? Is there existing software that implements the data, functions, and features that are required?

- *Has a similar problem been solved?* If so, are elements of the solution reusable?

- *Can subproblems be defined?* If so, are solutions readily apparent for the subproblems?

- *Can you represent a solution in a manner that leads to effective implementation?* Can a design model be created?

3.Carry Out the Plan

- *Does the solutions conform to the plan?* Is source code traceable to the design model?

- *Is each component part of the solution provably correct?* Has the design and code been reviewed, or better, have correctness proofs been applied to algorithm?

4. Examine the Result

- *Is it possible to test each component part of the solution?* Has a reasonable testing strategy been implemented?

- *Does the solution produce results that conform to the data, functions, and features that are required?* Has the software been validated against all stakeholder requirements?

2.3 Software Engineering Practice

2.3.2 Hooker's General Principles for SE Practice

1. *The Reason It All Exists: Value to users*
2. *KISS (Keep It Simple, Stupid!)*
3. *Maintain the Vision*
4. *What You Produce, Others Will Consume: Requirements produced will goto designer*
5. *Be Open to the Future*
6. *Plan Ahead for Reuse*
7. *Think!*