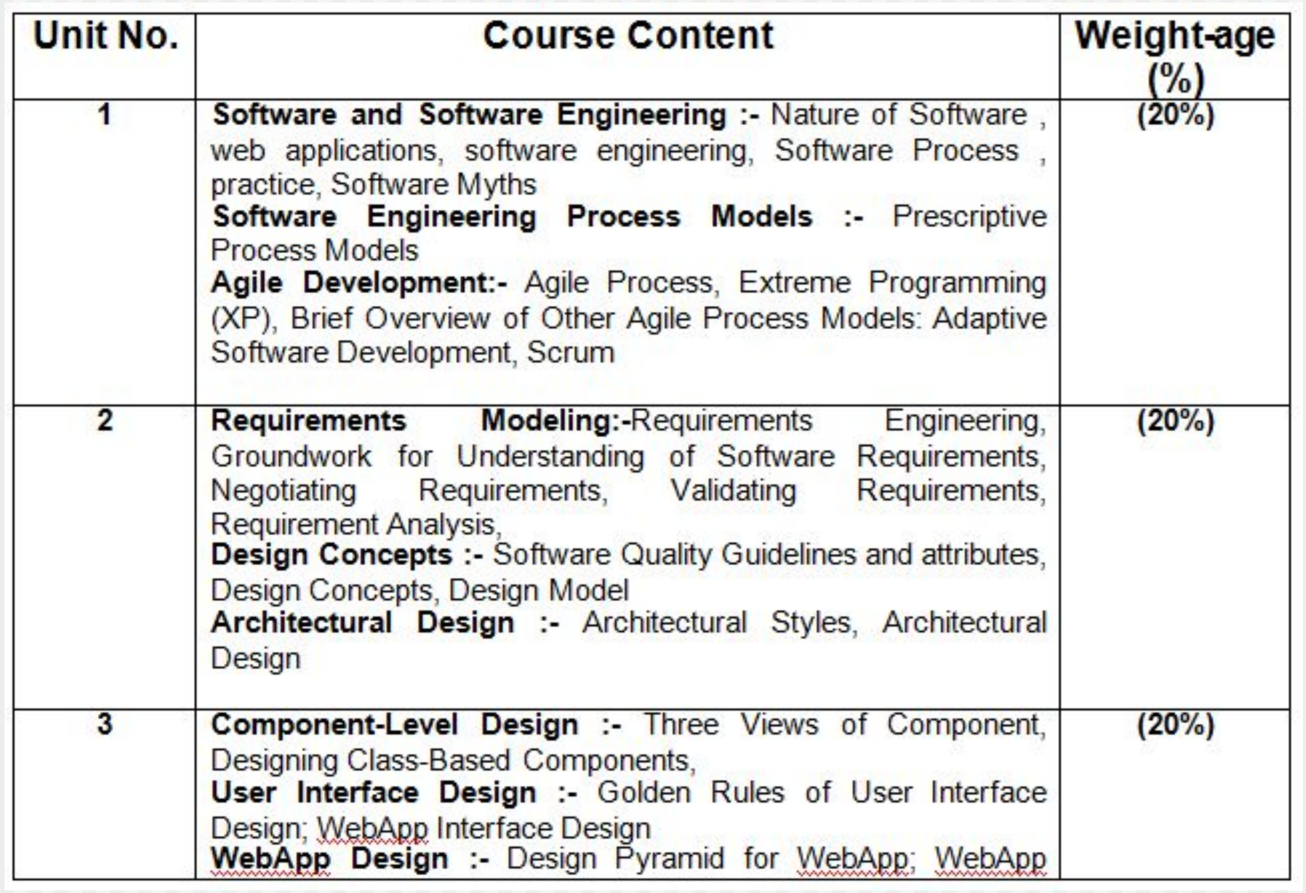
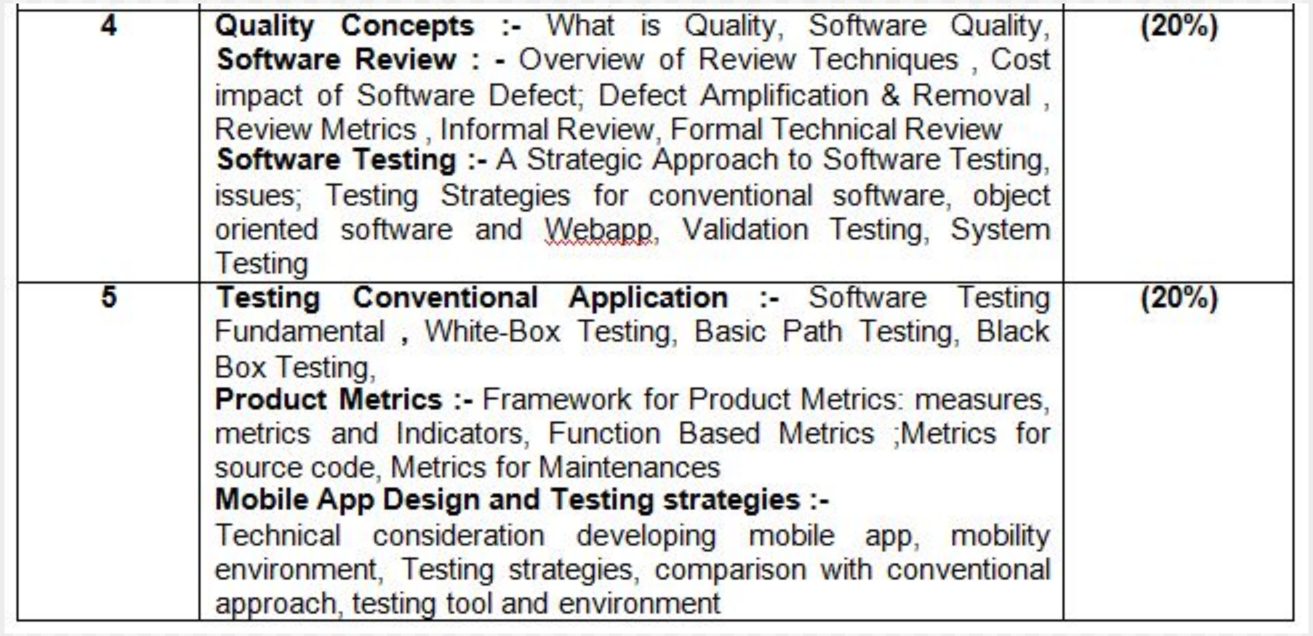
Software Engineering



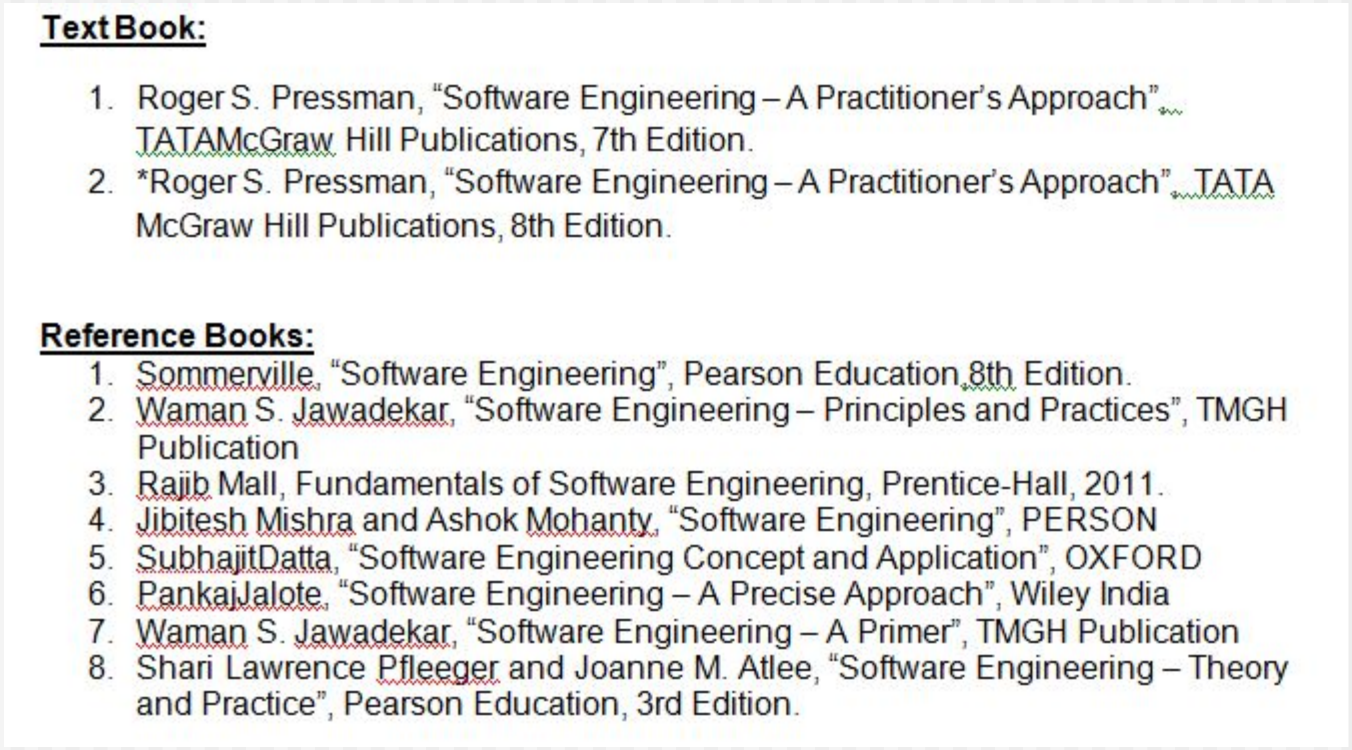
Unit No.	Course Content	Weight-age (%)
1	**Software and Software Engineering :-** Nature of Software , web applications, software engineering, Software Process , practice, Software Myths **Software Engineering Process Models :-** Prescriptive Process Models **Agile Development:-** Agile Process, Extreme Programming (XP), Brief Overview of Other Agile Process Models: Adaptive Software Development, Scrum	(20%)
2	**Requirements Modeling:-**Requirements Engineering, Groundwork for Understanding of Software Requirements, Negotiating Requirements, Validating Requirements, Requirement Analysis, **Design Concepts :-** Software Quality Guidelines and attributes, Design Concepts, Design Model **Architectural Design :-** Architectural Styles, Architectural Design	(20%)
3	**Component-Level Design :-** Three Views of Component, Designing Class-Based Components, **User Interface Design :-** Golden Rules of User Interface Design; WebApp Interface Design **WebApp Design :-** Design Pyramid for WebApp; WebApp	(20%)

Software Engineering



4	**Quality Concepts :-** What is Quality, Software Quality, **Software Review : -** Overview of Review Techniques , Cost impact of Software Defect; Defect Amplification & Removal , Review Metrics , Informal Review, Formal Technical Review **Software Testing :-** A Strategic Approach to Software Testing, issues; Testing Strategies for conventional software, object oriented software and Webapp, Validation Testing, System Testing	(20%)
5	**Testing Conventional Application :-** Software Testing Fundamental , White-Box Testing, Basic Path Testing, Black Box Testing, **Product Metrics :-** Framework for Product Metrics: measures, metrics and Indicators, Function Based Metrics ;Metrics for source code, Metrics for Maintenances **Mobile App Design and Testing strategies :-** Technical consideration developing mobile app, mobility environment, Testing strategies, comparison with conventional approach, testing tool and environment	(20%)

Software Engineering

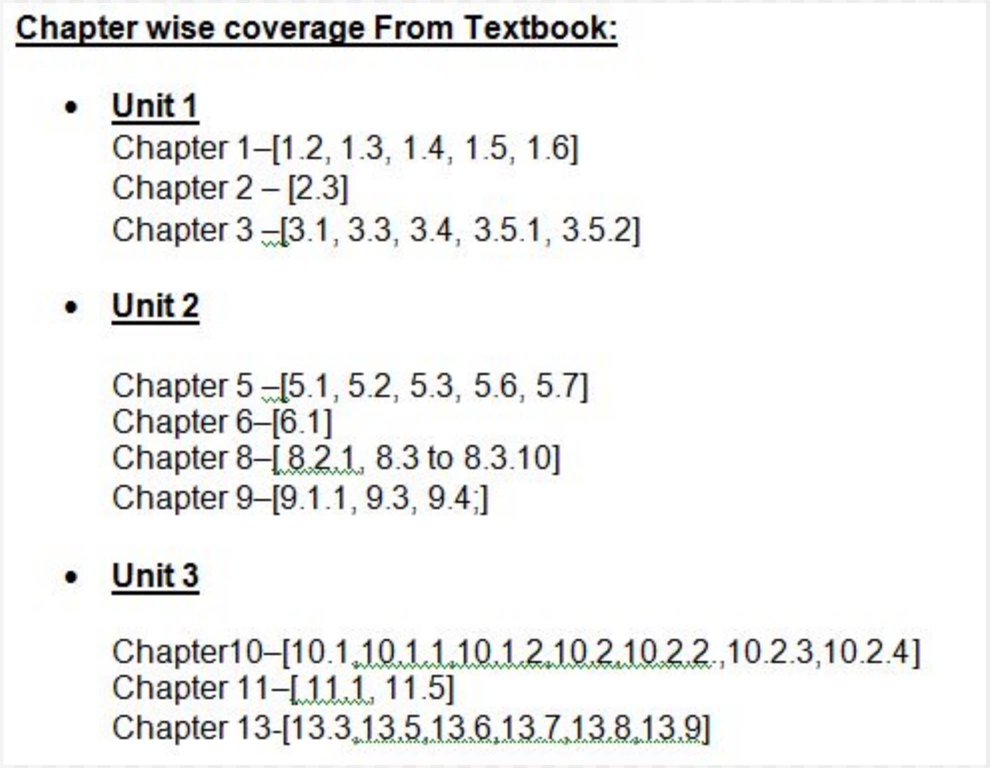


Text Book:

1. Roger S. Pressman, "Software Engineering – A Practitioner's Approach", TATAMcGraw Hill Publications, 7th Edition.
2. *Roger S. Pressman, "Software Engineering – A Practitioner's Approach", TATA McGraw Hill Publications, 8th Edition.

Reference Books:

1. Sommerville, "Software Engineering", Pearson Education,8th Edition.
2. Waman S. Jawadekar, "Software Engineering – Principles and Practices", TMGH Publication
3. Rajib Mall, Fundamentals of Software Engineering, Prentice-Hall, 2011.
4. Jibitesh Mishra and Ashok Mohanty, "Software Engineering", PERSON
5. SubhajitDatta, "Software Engineering Concept and Application", OXFORD
6. PankajJalote, "Software Engineering – A Precise Approach", Wiley India
7. Waman S. Jawadekar, "Software Engineering – A Primer", TMGH Publication
8. Shari Lawrence Pfleeger and Joanne M. Atlee, "Software Engineering – Theory and Practice", Pearson Education, 3rd Edition.



Chapter wise coverage From Textbook:

- **Unit 1**

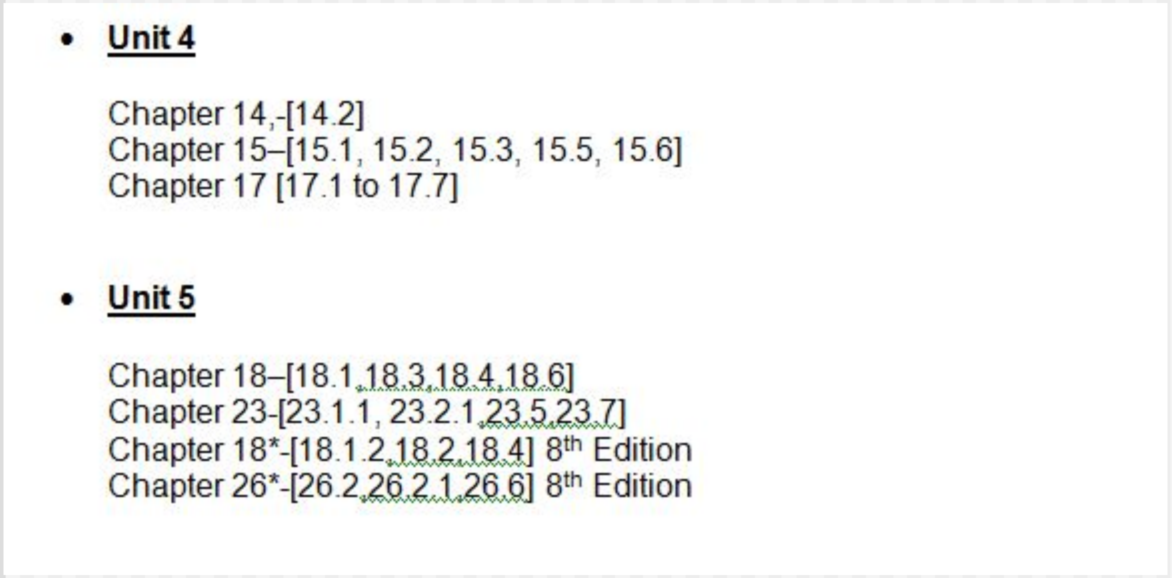
  Chapter 1–[1.2, 1.3, 1.4, 1.5, 1.6]
  Chapter 2 – [2.3]
  Chapter 3 –[3.1, 3.3, 3.4, 3.5.1, 3.5.2]

- **Unit 2**

  Chapter 5 –[5.1, 5.2, 5.3, 5.6, 5.7]
  Chapter 6–[6.1]
  Chapter 8–[8.2.1, 8.3 to 8.3.10]
  Chapter 9–[9.1.1, 9.3, 9.4;]

- **Unit 3**

  Chapter10–[10.1,10.1.1,10.1.2,10.2,10.2.2.,10.2.3,10.2.4]
  Chapter 11–[11.1, 11.5]
  Chapter 13-[13.3,13.5,13.6,13.7,13.8,13.9]



- **Unit 4**

  Chapter 14,-[14.2]
  Chapter 15–[15.1, 15.2, 15.3, 15.5, 15.6]
  Chapter 17 [17.1 to 17.7]

- **Unit 5**

  Chapter 18–[18.1,18.3,18.4,18.6]
  Chapter 23-[23.1.1, 23.2.1,23.5,23.7]
  Chapter 18*-[18.1.2,18.2,18.4] 8th Edition
  Chapter 26*-[26.2,26.2.1,26.6] 8th Edition

SOFTWARE
and
Software Engineering

Programming in the Small

- Individual Effort
- Not Fully Tested
- Used by the Developer Only
- Programming Language dependent
- Intellectual Exercise
- No Documentation
- Developer Maintenance (if at all)

Programming in the Large

- Team Effort
- Use of Methodology
- Documentation
- Schedule / Cost Control
- Quality Assurance
- Used by Untrained User
- Planning
- Use of Software Tools
- Conformance to Standards
- Reuse
- Non-Developer Maintenance
- Change Management
- Version Control
- Subject to Risks

Software

- *Software can be described as a collection of instruction that are executed to get desired functionalities. Software is made-up of*

1) instructions (programs) that when executed provide desired function and performance
2) data structures that enable the programs to adequately manipulate information
3) documents that describe the operation and use of the programs

Types of Software

- system software
- application software
- engineering/scientific software
- embedded software
- product-line software (word processing, spread sheet)
- WebApps (Web applications)
- AI software (solve better than human)

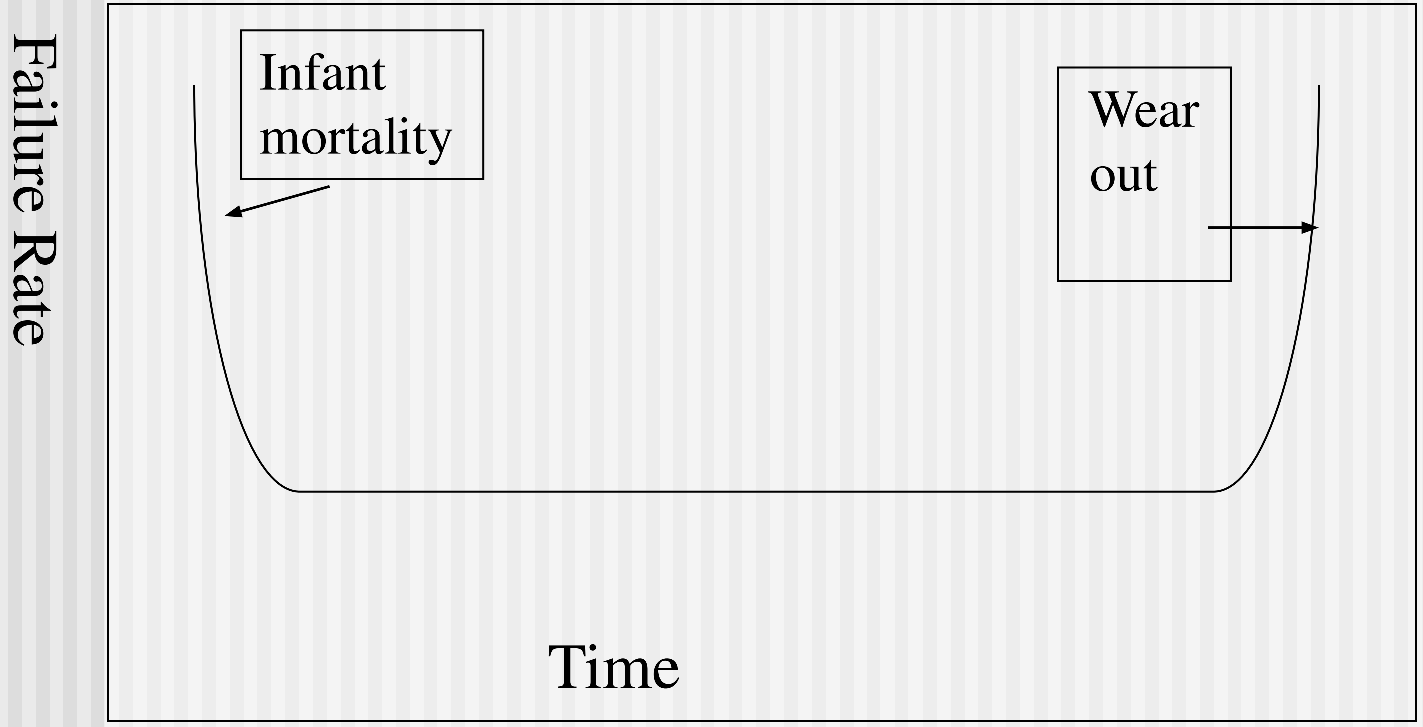
Software—New Categories

- Open world computing— universal, distributed computing due to wireless networks
  - New challenges for SE to develop S/W for mobile devices, enterprise system on vast network
- Netsourcing—the Web become a computing engine, content provider
  - New challenges for SE to architect simple and sophistical application to target word-wide end user
- Open source—”free” source code open to the computing community
- Also … (see Chapter 31)
  - Data mining & Grid computing
  - Cognitive machines(solve problem like human)
  - Software for nanotechnologies

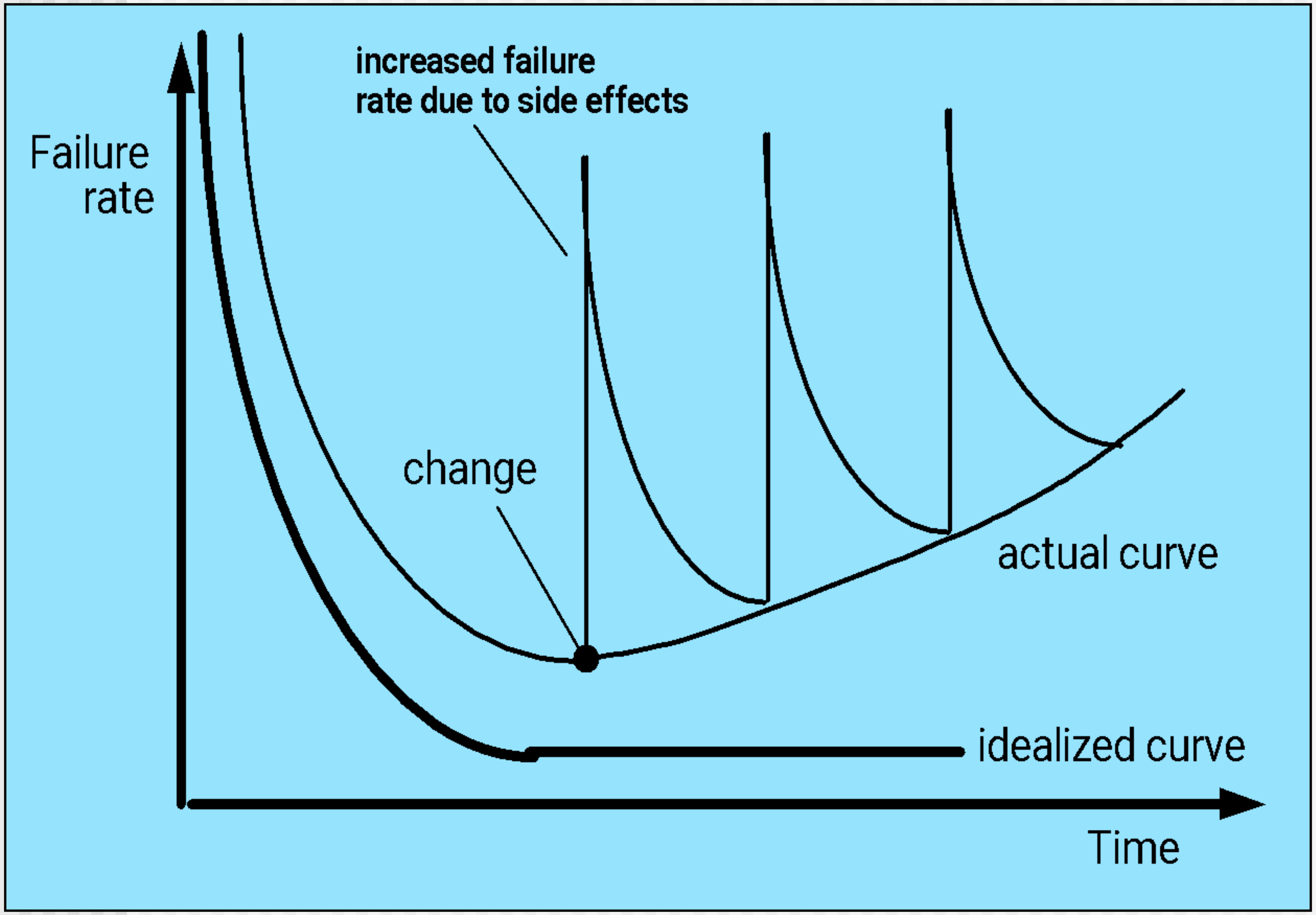
Characteristics of Software

- Not “Manufactured” in the Classical Sense
  - A logical rather than physical system element
- Invisibility
- Does Not “Wear-Out”
- Invariably Custom-Built
- Has reusable components
- Flexible

Failure ("Bathtub") Curve for Hardware



Wear vs. Deterioration



To Achieve Quality in Process and Product

- Identify steps to follow
- Create Model for the Involved Tasks
- Use Techniques to carry out Tasks
- Verify & Validate each Task and Results
- Need to apply Process & Project Management Skills
- Hence ……

  …………An "Engineering" Approach is essential

Software Engineering

IEEE Definition

"A Systematic Approach to the Development, Operation, Maintenance and Retirement of Software, where Software is Computer Programs, Procedures, Rules and Associated Documents and Data pertaining to the Operation of a Computer System."

*IEEE stands for Institute of Electrical and Electronics Engineers
Founder : American Institute of Electrical Engineers

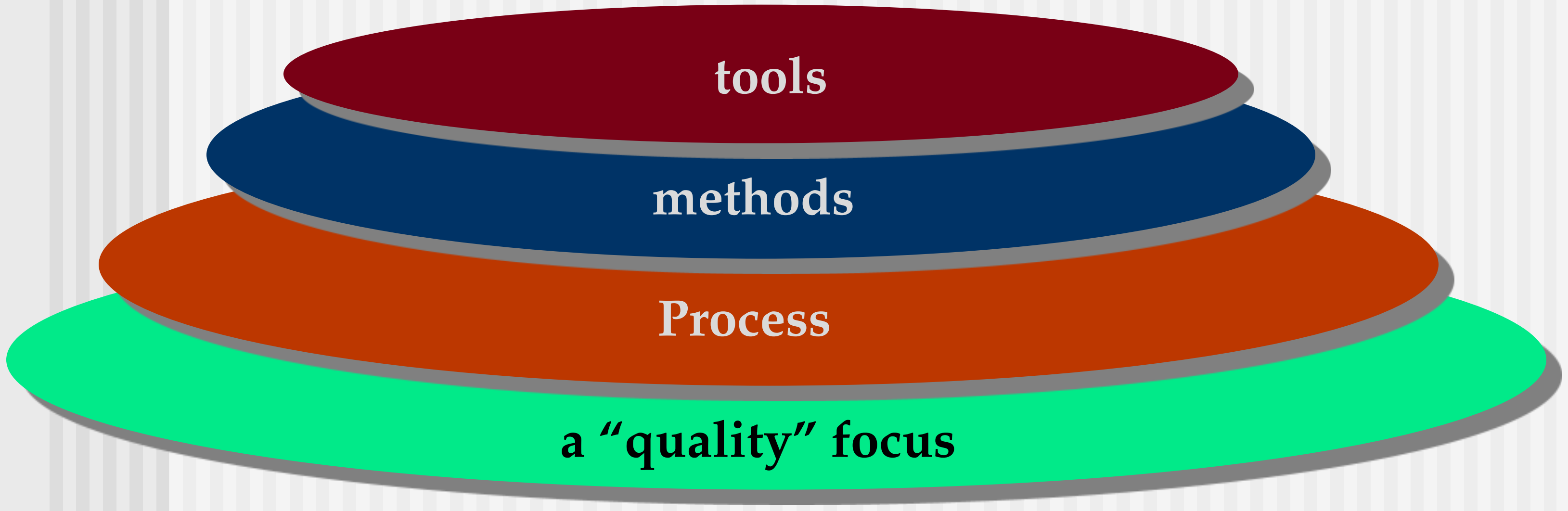
Software Engg involves…

- Computer Science Aspects
- Management Aspects
- Economical Aspects
- Psychological Aspects
- Software Engineer must be proficient in:
  - Tools
  - Testing
  - Project Management
  - Time Estimation
  - Coordination
  - Packaging
  - Teaming *and so on*

Software Engineering

- Some realities:
  - *an intensive effort should be made to understand the problem before a software solution is developed*
  - *design becomes a fundamental activity*
  - *software should exhibit high quality*
  - *software should be maintainable*
- The influential definition:
  - *[Software engineering is] the establishment and use of sound engineering principles in order to obtain economical software that is reliable and works efficiently on real machines.*

Software Engineering Layers



Software Engineering is a layered technology

S/W Engg. Layers

- Engg. Approach must rest on quality
- Process manage and control s/w projects and delivery of quality work products
- Method provide technical how-to's for communication, requirements analysis, design modeling, program construction, testing …
- Tools provide automated and semi automated support for the process and methods

The Software Process

- Software Process is the Collection of Activities ,Actions and Tasks , that are performed when some work product is to be created
  - Activities make every efforts to achieve a broad objective
  - Action encompasses a set of tasks that produce a major work product
  - A task focuses on a small well defined objectives to produce outcome
- A **process frame work** establishes the foundation for a complete software engineering process by identifying a small number of a framework activities , applicable to all software project , regardless of size, complexity

Process Framework Activities

- **Communication -** Communicate with stakeholders and customers to obtain objectives of the system and requirements for the software.
- **Planning -** Software project plan has details of resources needed, tasks and risk factors likely to occur, schedule.
- **Modeling -** Architectural models and design to better understand the problem and for working towards the best solution.
- **Construction -** Generation of code and testing of the system to rectify errors and ensuring all specified requirements are met.
- **Deployment -** Entire software product or partially completed product is delivered to customer for evaluation and feedback.

Umbrella Activities

Activities that occur throughout a software process for better management and tracking of the project.

- Software project tracking and control - Compare progress of the project with the plan and take steps to maintain planned schedule.
- Risk management - Evaluate risks that can affect the outcome and quality of the software product.
- Software quality assurance (SQA) - Conduct activities to ensure quality of the product.
- Technical reviews - Assessment of errors and correction done at each stage of activity.
- Measurement - All the measurements of project and product features.
- Software configuration management (SCM) - Controlling and tracking changes in the software.
- Reusability management - Back up work products for reuse and apply mechanism to achieve reusable software components.
- Work product preparation and production - Project planning and other activities used to create work product is documented.

How a Process Model for one project is different than other project ?

- the overall flow of activities, actions, and tasks and the interdependencies among them
- the degree to which actions and tasks are defined within each framework activity
- the degree to which work products are identified and required
- the manner which quality assurance activities are applied
- the manner in which project tracking and control activities are applied
- the overall degree of detail with which the process is described
- the degree to which the customer and other stakeholders are involved with the project
- the level of autonomy given to the software team
- the degree to which team organization and roles are prescribed

The Essence of Practice

- General steps in carrying out in software engineering are
    1. *Understand the problem* through communication.
    2. *Plan a solution* by modeling and software design.
    3. *Carry out the plan* for code generation.
    4. *Examine the result for accuracy* by testing and quality assurance.

Principle of best Practice

- Software should provide value to users.
- Keep the software design simple.
- Vision of the project should be kept in mind throughout.
- Keep your work products in an understandable format for others to work with them easily.
- Design in a way that can incorporate changes in future.
- Build reusable components.
- Think before acting or learn before starting.

Hooker’s General Principles

Focus on S/W engineering practice as a whole

- 1: *The Reason It All Exists : s/w system is required to , provide value to its user, so before starting any project ask “ Does this add real value to the system “ if answer is “no” , then do not do it*
- 2: *KISS (Keep It Simple, Stupid!) : so easy to maintain and generate less error*
- 3: *Maintain the Vision : without conceptual integrity , a system become patchwork of incompatible design*
- 4: *What You Produce, Others Will Consume*
- 5: *Be Open to the Future*
- 6: *Plan Ahead for Reuse*
- *7: Think! : placing clear, complete thought before action always produce better result*

Software Myths

- Affect managers, customers (and other non-technical stakeholders) and practitioners
- Are believable because they often have elements of truth,

but …

- Invariably lead to bad decisions,

therefore …

- Insist reality toward formulation of practical solution for software engineering

Software Myths

- Management
  - We have standards
  - We have new computers
  - We'll add more people to catch up
  - I outsourced it, I'm done
- Customer
  - We have general objectives, let's start
  - Change is easily accommodated
- Practitioner
  - We'll write it and be done
  - I can't assess quality until it is running
  - I only need deliver code
  - Software engineering is about meaningless documents

Software Myths

Management Myths

● Software practitioners follow the book of standards and procedures for software development.

● If a delay occurs, programmers can be added at any point.

● Outsourcing helps.

Customer Myths

● General requirements and not a detailed one is needed for starting programming.

● Software is easy to change.

Developer's Myths (Practitioner's Myths)

● Software developing is all about coding.

● Only a finished software can be tested.

● Software engineering has plentiful documentation and will slow down the process

How It all Starts

- Every software project is precipitated by some business need—
  - the need to correct a defect in an existing application;
  - the need to adapt a 'legacy system' to a changing business environment;
  - the need to extend the functions and features of an existing application, or
  - the need to create a new product, service, or system.

Exercise

1. Write down IEEE definition for Software engineering
2. List and explain characteristics of S/w
3. Explain with figure s/w failure / deterioration
4. Explain with figure s/w engineering layers
5. Define s/w process and explain process framework activities
6. Briefly explain Umbrella activities
7. Explain how a process model for one project is different than the other project
8. Explain in brief general steps in carrying out in s/w engineering
9. Explain principle of best practice / or Hooker's general principles
10. Explain S/w Myths