Syllabus

Module	Topics	No. of Lectures
1	Software and Software Engineering, Process Models, Agile Development, Principles that Guide Practice, Understanding requirements, Requirements modeling: scenarios, information, and analysis classes	9
2	Requirements modeling: flow, behavior, patterns, and webapps, Design Concepts, Architectural design, Component-level Design, User Interface Design	8
3	Pattern-Based Design, WebApp Design, Quality Concepts, Review Techniques, Software Quality Assurance, Software Testing Strategies, Testing Conventional Applications	9
4	Testing Object-Oriented Applications, Testing Web Applications, Formal Modeling and Verification, Software Configuration Management, Product Metrics, Project Management Concepts, Process and Project Metrics	8
5	Estimation for Software Projects, Project Scheduling, Risk Management, Maintenance and Reengineering, Software Process Improvement, Emerging Trends in Software Engineering, Concluding Comments	8

Course Details

L-T-P: 3-0-0

Weightage

- Mid-Sem: 30
- End-Sem: 50
- Quiz: 20

Textbook: Software Engineering A Practitioner's Approach 7th Edition Roger S. Pressman

Chapter 1: Software and Software Engineering

Table of Contents

Part 01　Introduction & Objectives

Lecture Objectives

Define software, its evolving nature, and its various application domains.

Software is more than just programs; encompassing programs, documentation, operating procedures, & data, evolving across various application domains.

Define Software Engineering and understand its layered process model.

Software engineering is the application of a systematic, disciplined approach to the development, operation, and maintenance of software, and understand its layered process model.

Distinguish conventional software from Web Applications.

Web applications exhibit unique attributes, distinct from conventional software, including network intensity, concurrency, and continuous evolution.

Identify core software engineering practices and dispel common software myths.

Core practices include understanding the problem, planning a solution, carrying out the plan, and examining the result, dispelling common software myths.

Part 02 The Nature of Software

Defining Software

Software Components

Software consists of programs, documentation, operating procedures, and data, highlighting its logical, not physical, nature, being developed rather than manufactured.

Key Software Characteristics

Software is developed through careful design, doesn't wear out but can deteriorate with changes, and is often custom-built, though component reuse is increasing.

FIGURE 1.1

Failure curve for hardware

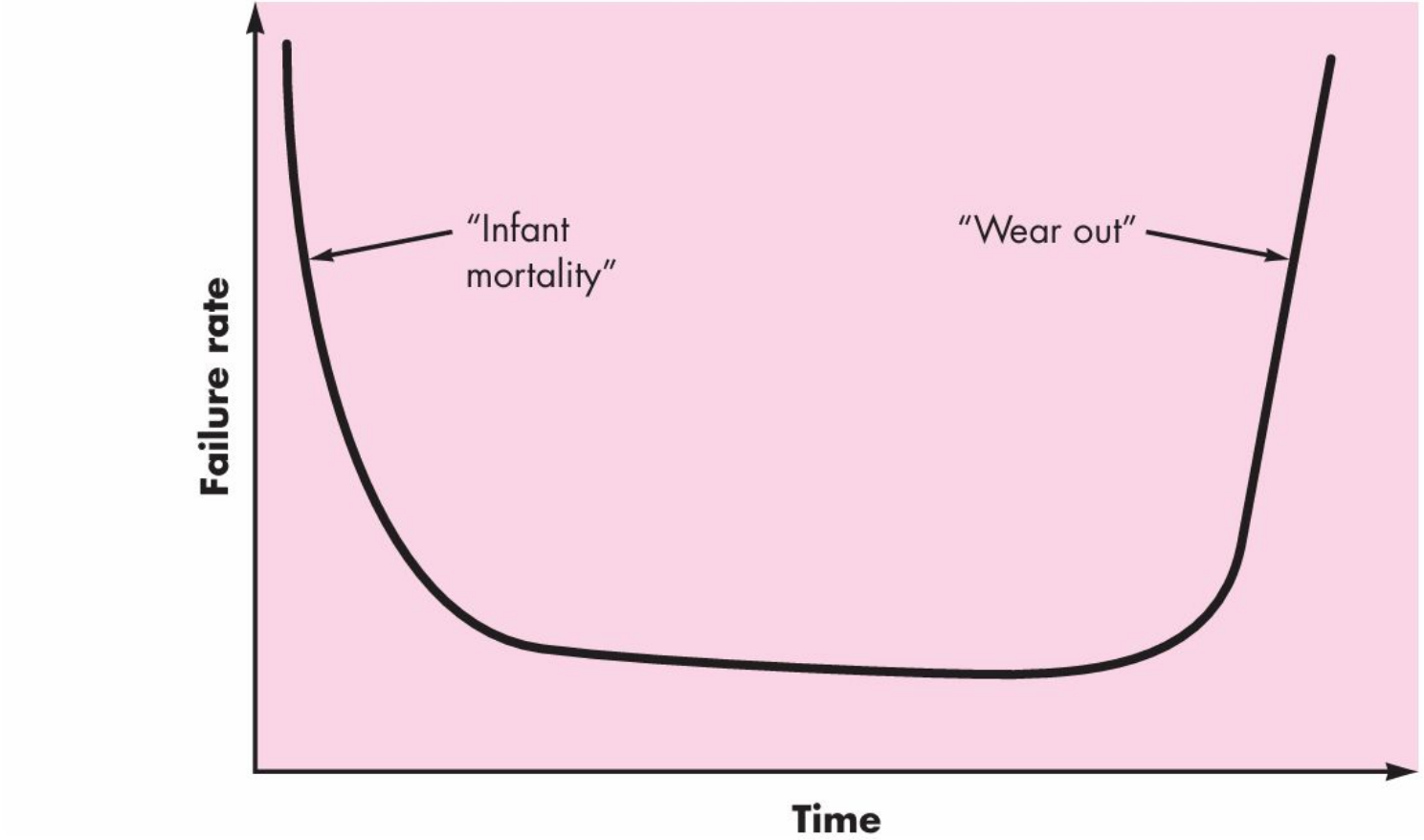
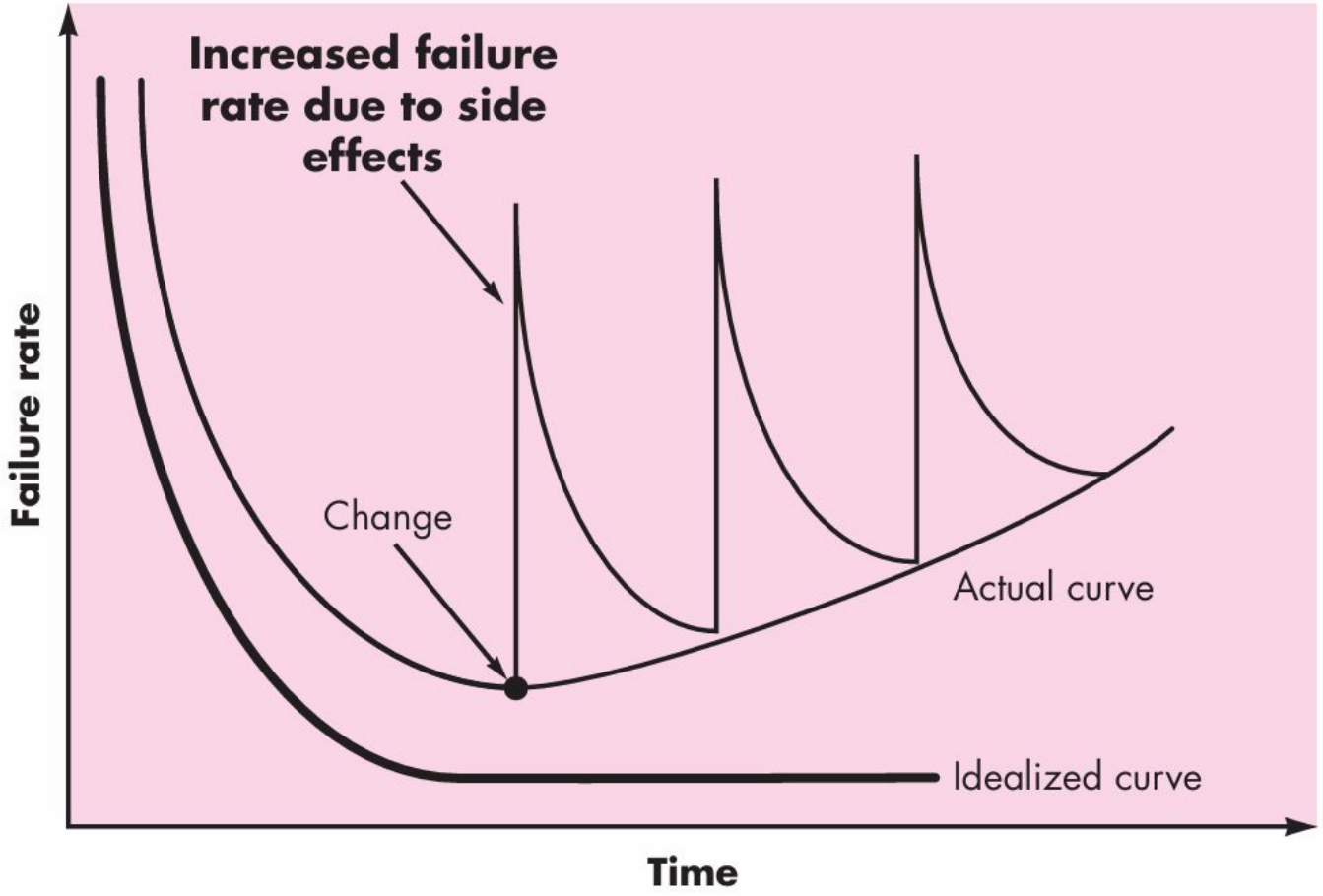


FIGURE 1.2

Failure curves for software



Software Application Domains

System Software

System software includes operating systems, compilers, and drivers like Windows and Linux kernels, managing computer resources and providing core services.

Application Software

Application software provides standalone programs for specific business needs, such as word processors, photo editors, and enterprise resource planning systems.

Engineering/Scientific Software

Engineering and scientific software is used for simulation, modeling, and data analysis, with examples like MATLAB and ANSYS enhancing research capabilities.

Embedded Software

Embedded software resides in read-only memory, controlling products and systems, ranging from firmware in microwaves to car engine controllers for real-time operations.

Artificial Intelligence Software

Artificial intelligence software uses algorithms for non-numerical problem-solving, including expert systems, robotics, and neural networks, mimicking human cognitive functions.

Web and Mobile Applications

Web and mobile applications are network-centric, browser-accessible, emphasizing continuous deployment, high aesthetics, and critical security across online platforms.

Product-Line Software

Product-line software is designed to provide specific capabilities for many customers, such as ERP and CRM systems like Salesforce, supporting diverse organizational functions.

Legacy Software

Definition and Challenges

Legacy software is older software vital to an organization, often difficult to maintain due to outdated methods, poor documentation, and architectural limitations.

The Engineering Challenge

The challenge lies in evolutionary maintenance, adapting legacy systems to meet new needs and integrate with modern systems, ensuring continued operational efficiency.

Part 03 The Unique Nature of WebApps

Web Apps: A Distinct Category

Key Attributes of Web Apps

Web apps operate over networks, handle concurrency, face unpredictable loads, undergo continuous evolution, prioritize aesthetics, are content-sensitive, and demand high security.

Implications for Development

These attributes necessitate specific process models like Agile and DevOps, emphasizing performance, security, and usability in modern web development practices.

Part 04 Software Engineering - The Discipline

Software Engineering: A Formal Definition

IEEE Definition

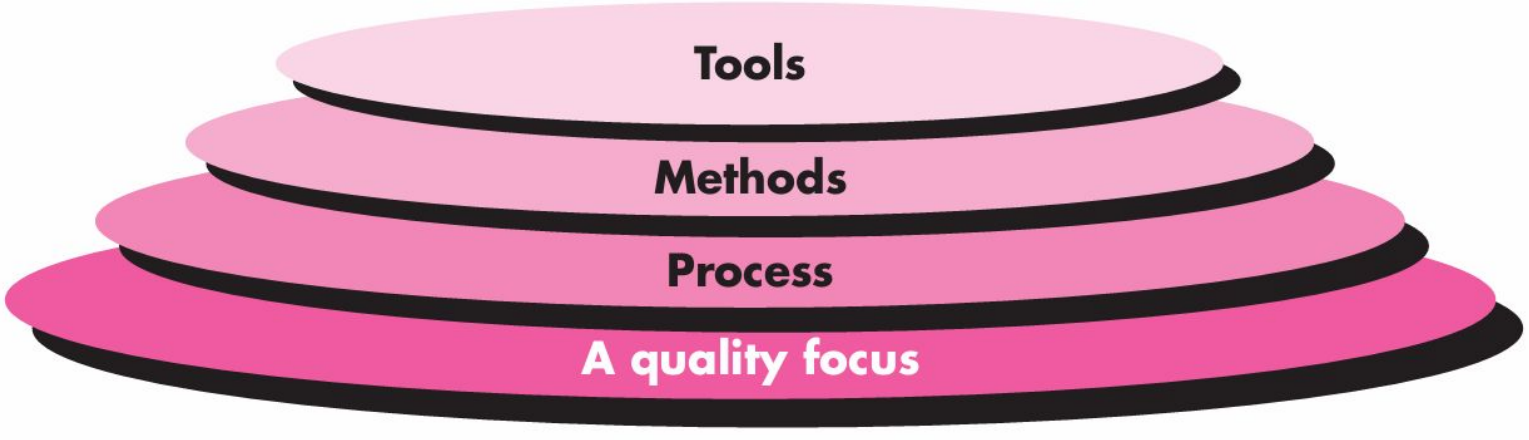
The IEEE defines software engineering as applying engineering principles to software development, operation, and maintenance, emphasizing a systematic, disciplined approach.

Key Goal of Software Engineering

The key goal is to produce high-quality software on time and within budget, satisfying users' evolving needs, thereby enhancing business value and customer satisfaction.

FIGURE 1.3

Software engineering layers



The Software Process

Software Process Definition

A framework of tasks required to build high-quality software, providing a roadmap with a sequence of steps applicable to all projects, ensuring structured development.

Common Process Framework Activities

Activities include communication, planning, modeling, construction, and deployment, covering stakeholder collaboration, project scheduling, design modeling, coding-testing and feedback collection.

Software Engineering Practice

The Essence of Practice

Practice bridges abstract principles and actual creation, involving understanding the problem, planning a solution, carrying out the plan, and examining the result.

General Principles

Understand the problem thoroughly, plan before designing, prioritize quality, maintain a clear vision, and think before acting to ensure efficient and effective engineering.

Part 05 Software Myths & How It All Starts

Software Myths

01

Management Myths

Myths include assuming standards ensure a process, adding programmers catches up, and outsourcing requires no management, leading to project mismanagement.

02

Customer Myths

Myths include vague objectives suffice and requirements can change anytime easily, causing project failures due to unclear scope and costly changes.

03

Practitioner's Myths

Myths include coding immediately saves time, the job ends with a working program, and only the program is deliverable, resulting in rework and maintenance issues.

How It All Starts

01

The Seed of an Idea

Projects begin with a need, opportunity, or market need, prompting the recognition of a software solution to address the identified gap or demand.

02

The First Engineering Step

Stakeholders meet with software engineers to transform vague concepts into actionable scope statements and preliminary requirements, transforming into actionable development.

Part 06 Conclusion & Key Takeaways

Software Complexity and Engineering

Software as a Complex Construct

Software is a complex, logical construct across many domains, with WebApps presenting unique challenges, requiring specialized development approaches.

The Importance of Software Engineering

Software engineering is the disciplined, layered response to building reliable software, guided by understanding, planning, quality, and awareness of common myths.

Thank You!!