GANPAT UNIVERSITY

U. V. PATEL COLLEGE OF ENGINEERING

2CEIT502

SOFTWARE ENGINEERING

LECTURE 1

INTRODUCTION TO SOFTWARE AND SOFTWARE ENGINEERING

Prepared by: Prof. Ravi Raval (Asst. Prof in C.E Dept. , UVPCE)

Course Contents

Units to be covered by RFR

1. Introduction to software and software engineering
2. Process Models
3. Building the analysis models
4. Requirement analysis and requirement specification
5. Software Project Management
6. Coding and testing

Units to be covered by YJP

7. Software Design
8. Unified Modeling Language
9. Function oriented software design
10. Advanced topics in software engineering

Books:

(1) Software Engineering a practitioner's approach by Roger S. Pressman

(2) Fundamentals of Software Engineering by Rajib Mal

Teaching and Examination marks Scheme

Teaching Scheme

Theory	Practical
3	1

Theory

Internal Exam (Best out of 2)	Quiz (Best out of 2)	Assignments	Lecture Attendance	Final Theory Exam	**Total**
20	10	5	5	60	**100**

Practical

Continuous Evaluation	Lab Attendance	Term-work / File	Practical Exam	Viva voce	**Total**
15	5	10	10	10	**50**

Unit 1: Intro. to Software and Software Engineering

Contents:

- FAQ about Software Engineering
- Software characteristics,
- The Changing Nature of Software
- Software Myths

Before discussing contents:

- Why we are learning SE?
- What is the significance of SE in Computer Science / IT?
- What is software?
- What is software Engineering?
- What are the different application domains of a software?

Unit 1: Intro. to Software and Software Engineering

Why we are learning Software Engineering?

- Reduce Complexity
- To minimize software cost
- To decrease time
- Handling big projects
- Reliable softwares
- Effectiveness

What is the significance of SE in Computer Science / IT?

- Difference between Script Vs. Program Vs. Software

- Script is interpreted vs. program is executed.
- *A "script" is code that acts upon some system in an external or independent manner* and can be removed or disabled without disabling the system itself.
- *A "program" is code that constitutes a system.*
- **A set of program makes up a software**

Unit 1: Intro. to Software and Software Engineering

What is software?

Software is:

(1) instructions (computer programs) that when executed provide desired features, function, and performance;

(2) data structures that enable the programs to adequately manipulate information, and

(3) descriptive information in both hard copy and virtual forms that describes the operation and use of the programs.

- Software is a logical rather than a physical system element. Therefore, software has characteristics that are considerably different than those of hardware:

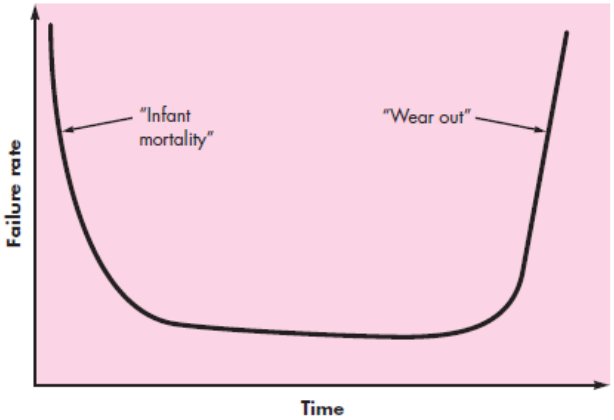
Unit 1: Intro. to Software and Software Engineering

1. **Software is developed or engineered; It is not manufactured in the classical sense.**

In both activities, high quality is achieved through good design, but the manufacturing phase for hardware can introduce quality problems that are nonexistent (or easily corrected) for software. Both activities are dependent on people, but the relationship between people applied and work accomplished is entirely different. Both activities require the construction of a "product," but the approaches are different

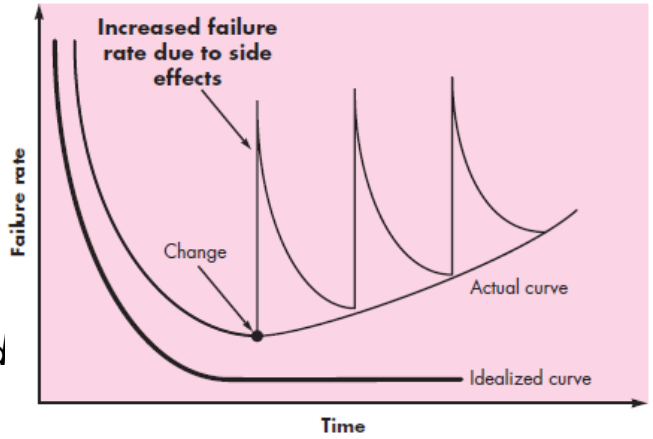
2. **Software doesn't wear out.**

Hardware failure rate rises as components suffers from cumulative effects of dust, vibration, abuse, temperature humidity etc. Software is not susceptible to the environment maladies



Unit 1: Intro. to Software and Software Engineering

The failure rate curve for software should take the form of the "idealized curve" shown in Figure. Undiscovered defects will cause high failure rates early in the life of a program.



3. *Although the industry is moving toward component-based construction, most softwares continues to be custom built.*

Software Application Domains

(1) System Software (2) Application software (3) Engineering / Scientific softwares (4) Embedded Softwares (5) Product-line softwares (6) Web applications (7) AI Applications (8) Open world computing (9) Netsourcing (10) Open source.

Unit 1: Intro. to Software and Software Engineering

Unique Nature of Web Apps:

1. **Network intensiveness:** must serve diverse community of clients
2. **Concurrency:** large no. of users simultaneously
3. **Unpredictable load:** week days/ weekend
4. **Performance:** must not wait too long
5. **Availability:** 24x7x365
6. **Data driven:** text, graphics, audio, video
7. **Content sensitive:**
8. **Continuous evolutions:** regular updates
9. **Immediacy:** compelling need software markets more
10. **Security:**
11. **Aesthetics:** look and feel.

Unit 1: Intro. to Software and Software Engineering

What is software engineering?

Before developing a software; you should understand few realities of softwares:

1. It follows that a concerted effort should be made to understand the problem before a software solution is developed.
2. It follows that design becomes a pivotal activity.
3. It follows that software should exhibit high quality.
4. *It follows that software should be maintainable.*

"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." - **Fritz Bauer [Nau69]**

"The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software;" - **IEEE [IEE93a]**

Unit 1: Intro. to Software and Software Engineering

FAQs about Software Engineering (ref. Sommeville 9th Ed):

Questions	Answers
What is software?	Computer programs and associated documentation. Software products may be developed for a particular customer or may be developed for a general market.
What are the attributes of good software?	Good software should deliver the required functionality and performance to the user and should be maintainable, dependable, and usable.
What is software engineering?	Software engineering is an engineering discipline that is concerned with all aspects of software production.
What are the fundamental software engineering activities?	Software specification, software development, software validation, and software evolution.
What is the difference between software engineering and computer science?	Computer science focuses on theory and fundamentals; software engineering is concerned with the practicalities of developing and delivering useful software.

Unit 1: Intro. to Software and Software Engineering

FAQs about Software Engineering (ref. Sommeville 9th Ed) (cont.):

Questions	Answers
What is the difference between software engineering and system engineering?	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
What are the key challenges facing software engineering?	Coping with increasing diversity, demands for reduced delivery times, and developing trustworthy software
What are the costs of software engineering?	Roughly 60% of software costs are development costs; 40% are testing costs. For custom software, evolution costs often exceed development costs.
What are the best software engineering techniques and methods?	While all software projects have to be professionally managed and developed, different techniques are appropriate for different types of system. For example, games should always be developed using a series of prototypes whereas safety critical control systems require a complete and analyzable specification to be developed. You can't, therefore, say that one method is better than another.

Unit 1: Intro. to Software and Software Engineering

FAQs about Software Engineering (ref. Sommeville 9th Ed) (cont.):

Questions	Answers
What differences has the Web made to software engineering?	The Web has led to the availability of software services and the possibility of developing highly distributed service-based systems. Web-based systems development has led to important advances in programming languages and software reuse.

Unit 1: Intro. to Software and Software Engineering

Characteristics/Attributes of a good Software:

Product characteristics	Description
Maintainability	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.
Dependability and security	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.
Efficiency	Software should not make wasteful use of system resources such as memory and processor cycles. Efficiency therefore includes responsiveness, processing time, memory utilization, etc.
Acceptability	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.

Unit 1: Intro. to Software and Software Engineering

The changing nature of the softwares:

there are three general issues that affect many different types of software:

(1) Heterogeneity (2) Business & social change (3) Security & Trust

And hence we have diversity in software engineering like:

(1) Stand alone application
(2) Interactive transaction-based applications
(3) Embedded Control System
(4) Batch Processing system
(5) Entertainment System
(6) System for modeling and simulations
(7) Data collection system
(8) System of systems

Unit 1: Intro. to Software and Software Engineering

Software Myths: Myths that means believed to be TRUE but actually it's NOT

Management Myths		
Myth: We already have a book that's full of standards and procedures for building software. Won't that provide my people with everything they need to know?	**Myth:** If we get behind schedule, we can add more programmers and catch up (sometimes called the "Mongolian horde" concept).	**Myth:** If I decide to outsource the software project to a third party, I can just relax and let that firm build it.
Reality: The book of standards may very well exist, but is it used? Are software practitioners aware of its existence? Does it reflect modern software engineering practice? Is it complete? Is it adaptable? In many cases, the answer to all of these questions is "NO."	**Reality:** Software development is not a mechanistic process like manufacturing. new people are added, people who were working must spend time educating the newcomers, thereby reducing the amount of time spent on productive development effort.	**Reality:** If an organization does not understand how to manage and control software projects internally, it will invariably struggle when it outsources software projects.

Unit 1: Intro. to Software and Software Engineering

Software Myths: Myths that means believed to be TRUE but actually it's NOT

Customer Myths	
Myth: A general statement of objectives is sufficient to begin writing programs—we can fill in the details later.	**Myth:** Software requirements continually change, but change can be easily accommodated because software is flexible.
Reality: Although a comprehensive and stable statement of requirements is not always possible, an ambiguous "statement of objectives" is a recipe for disaster. Unambiguous requirements (usually derived iteratively) are developed only through effective and continuous communication between customer and developer.	**Reality:** It is true that software requirements change, but the impact of change varies with the time at which it is introduced. When requirements changes are requested early (before design or code has been started), the cost impact is relatively small. However, as time passes, the cost impact grows rapidly—resources have been committed, a design framework has been established, and change can cause upheaval that requires additional resources and major design modification.

Unit 1: Intro. to Software and Software Engineering

Software Myths: Myths that means believed to be TRUE but actually it's NOT

Developers Myths	
Myth: Once we write the program and get it to work, our job is done.	**Myth:** Until I get the program "running" I have no way of assessing its quality.
Reality: Someone once said that "the sooner you begin 'writing code,' the longer it'll take you to get done." Industry data indicate that between 60 and 80 percent of all effort expended on software will be expended after it is delivered to the customer for the first time.	**Reality:** One of the most effective software quality assurance mechanisms can be applied from the inception of a project—the technical review. Software reviews are a "quality filter" that have been found to be more effective than testing for finding certain classes of software defects.
Myth: The only deliverable work product for a successful project is the working program.	**Myth:** Software engineering will make us create voluminous and unnecessary documentation and which slow us down.
Reality: A working program is only one part of a software configuration that includes many elements. A variety of work products (e.g., models, documents, plans) provide a foundation for successful engineering, guidance and support.	Reality: Software engineering is not about creating documents. It is about creating a quality product. Better quality leads to reduced rework. And reduced rework results in faster delivery times.


THANK YOU!
ANY QUESTIONS?