GUDLAVALLERU ENGINEERING COLLEGE

(An Autonomous Institute with Permanent Affiliation to JNTUK, Kakinada)

Seshadri Rao Knowledge Village, Gudlavalleru – 521 356.

Department of Computer Science and Engineering

HANDOUT

on

SOFTWARE ENGINEERING

Vision

To be a Centre of Excellence in computer science and engineering education and training to meet the challenging needs of the industry and society.

Mission

- To impart quality education through well-designed curriculum in tune with the growing software needs of the industry.
- To serve our students by inculcating in them problem solving, leadership, teamwork skills and the value of commitment to quality, ethical behavior & respect for others.
- To foster industry-academia relationship for mutual benefit and growth.

Program Educational Objectives

- Identify, analyze, formulate and solve Computer Science and Engineering problems both independently and in a team environment by using the appropriate modern tools.
- Manage software projects with significant technical, legal, ethical, social, environmental and economical considerations.
- Demonstrate commitment and progress in lifelong learning, professional development, and leadership and communicate effectively with professional clients and the public.

HANDOUT ON SOFTWARE ENGINEERING

Class& Sem. : III B.Tech – II Semester Year : 2018-19

Branch : CSE Credits : 3

==========

1. Brief History and Scope of the Subject

Software engineering is the branch of computer science that creates practical, cost-effective solutions to computing and information processing problems, preferentially by applying scientific knowledge, developing software systems in the service of mankind. This course covers the fundamentals of software engineering, including understanding system requirements, finding appropriate engineering compromises, effective methods of design, coding, and testing, team software development, and the application of engineering tools. The course will combine a strong technical focus with a capstone project providing the opportunity to practice engineering knowledge, skills, and practices in a realistic development setting with a real client.

2. Pre-Requisites

- Familiar with the fundamental concepts of computers.

3. Course Objectives:

- Illustrate basic taxonomy and terminology of the software engineering.
- Plan and monitor the control aspects of project.

4. Course Outcomes:

Upon successful completion of the course, the students will be able to

CO1: explain the basic concepts of Software Engineering.

CO2: select the suitable process model based on the client requirements.

CO3: calculate software proficiency in terms of cost and schedule.

CO4: list the specifications of end-user according to business needs.

CO5: choose the appropriate architectural style for a given Scenario.

CO6: infer the system model for a sample case study.

CO7: deduce test cases by following different testing methodologies.

CO8: Explore the basic concepts of software engineering.

5. Program Outcomes:

Graduates of the Computer Science and Engineering Program will have

a) an ability to apply knowledge of mathematics, science, and engineering

b) an ability to design and conduct experiments, as well as to analyze and interpret data

c) an ability to design a system, component, or process to meet desired needs within realistic constraints such as economic, environmental, social, political, ethical, health and safety, manufacturability, and sustainability

d) an ability to function on multidisciplinary teams

e) an ability to identify, formulate, and solve engineering problems

f) an understanding of professional and ethical responsibility

g) an ability to communicate effectively

h) the broad education necessary to understand the impact of engineering solutions in a global, economic, environmental, and societal context

i) a recognition of the need for, and an ability to engage in life-long learning,

j) a knowledge of contemporary issues

k) an ability to use the techniques, skills, and modern engineering tools necessary for engineering practice.

6. Mapping of Course Outcomes with Program Outcomes:

	a	B	c	d	e	f	G	h	i	j	k
CO1							**H**				
CO2		**M**									
CO3					**M**						
CO4		**M**									
CO5										**L**	
CO6											
CO7											
CO8											

7. Prescribed Text Books

a. Pankaj Jalote, "A Concise Introduction to Software Engineering", Springer International Edition.

b. Roger S. Pressman, "Software Engineering", 7th edition, TMH.

8. Reference Text Books

a. K.K Aggarwal and Yogesh Singh, "Software Engineering", 3rd Edition, New Age Publications.

b. Sommerville, "Software Engineering", 8th edition, Pearson.

9. URLs and Other E-Learning Resources

a. https://www.learningware.in

b. http://www.learnerstv.com/engineering.php

c. http:/www.mhhe.com/pressman

d. http:/www.software-engin.com

e. http:/www.sei.cmu.edu

f. http:/www.scitools.com

g. http:/www.galorath.com

10. Digital Learning Materials:

- https://onlinecourses.nptel.ac.in

11. Lecture Schedule / Lesson Plan

Topic	No. of Periods	
	Theory	Tutorial
UNIT –1: Introduction to Software Engineering		
The evolving role of software	1	1
Changing nature of software	2	
Software myths	2	
The software problem: cost, schedule and quality	2	1
Scale and change	1	
	8	**2**
UNIT – 2: Software Process		
Process and project	1	1
Software development process models: waterfall model	2	
Prototyping , Iterative development	2	1
Relational unified process, Extreme programming and agile process.	3	
	8	**2**
UNIT – 3: Planning a software project		
Effort estimation	2	1
Project schedule and staffing	2	
Quality planning	2	1
risk management planning	2	
	8	**2**

UNIT – 4: Software requirement analysis and specification		
Introduction, Value of good SRS	2	1
Requirement process, Requirement specification	3	1
functional specification with use cases	3	
	8	**2**
UNIT – 5: Software Architecture and Design		
Role of software architecture, architecture views	2	2
Components and connector view, architecture styles for C & C view	3	
Function-oriented design	2	
Object oriented design	2	1
Metrics for design	2	
	11	**3**
UNIT – 6: Coding and Unit testing		
Programming principles and guidelines	2	1
Testing concepts, testing process	2	
Black-box testing, white-box testing	3	2
Metrics for testing	2	
	9	**3**
Total No. of Periods:	**52**	**14**

12. Seminar Topics

- Eye Tracking Software
- Agile Supply Chain
- Reconfigurable Manufacturing System
- Micro Air Vehicle
- Adhoc Wireless Networks
- Software Testing
- Liquid Lens
- Monorail
- Artificial Eye
- Biometric Voting System
- Infrared Plastic Solar Cell
- Solar Mobile Charger

Unit – I

INTRODUCTION TO SOFTWARE ENGINEERING

1. The Evolving role of software

- The role of computer software has undergone significant change over the last 50 years.
- Dramatic improvements in hardware performance, profound changes in computing architectures, vast increases in memory and storage capacity, and a wide variety of exotic input and output options have lead to the development of sophisticated and complex computer-based systems.
- Popular books published during the 1970s and 1980s described the changing nature of computers and software and their impact on our culture.
- Some of them stated that computers and software
    - Caused new industrial revolution
    - Lead to a transformation from an industrial society to an information society
    - Are the key to knowledge interchange throughout the world
- As the 1990s began, computers and software lead to a democratization of knowledge.
- During the later 1990s, the internet became very popular and lead to the development of web-based software systems. During this time many sectors like banking, insurance, airlines etc. have automated.
- Today, ubiquitous computing has created a generation of information appliances that have connectivity to the Web to provide a blanket of connectedness over our homes, offices and motorways.
- Software's role continues to expand still.

1.2 Software

- Software is a set of

  Instructions that when executed provide desired function and performance,

Data Structures that enable the programs to manipulate information,

Documents that describe the operation and use of the programs.

- Software is a logical rather than a physical system element.

1.2.1 Characteristics of Software

Every software exhibits three kids of characteristics.

1. *Software is developed or engineered, it is not manufactured*
   - Hardware is manufactured, but software is developed.
   - Both activities require the construction of a product but the approaches are different.
2. *Software doesn't "wear out"*
   - Environmental problems such as dust, vibration, abuse, temperature extremes etc. may cause the hardware to wear out. On the other hand, environmental problems can't influence the software and therefore it does not wear out.
   - Figure 1.1 depicts failure rate as a function of time for hardware. The relationship is often called the "bathtub curve". It indicates that hardware exhibits relatively high failure rates early in its life.
   - These failures are often due to design or manufacturing defects.
   - These defects can be corrected and the failure rate drops to a steady-state level (ideally, quite low) for some period of time.
   - As time passes, however, the failure rate rises again as hardware components suffer from the cumulative effects of dust, vibration, abuse, temperature extremes, and many other environmental maladies, which means that the hardware begins to wear out.

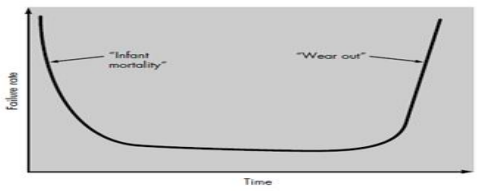


Figure 1.1

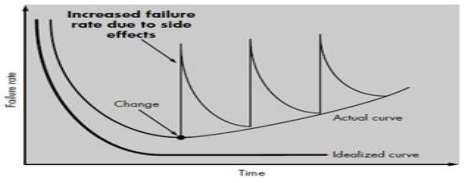


Figure 1.2

- Figure 1.2 depicts failure rate curve of software and it takes the form of "idealized curve".

- Undiscovered defects will cause high failure rates early in the life of a program. However, these are corrected and the curve flattens as shown in Figure 1.2.
- During its life, software will undergo change (maintenance).
- As changes are made, it is likely that some new defects will be introduced, causing the failure rate curve to spike as shown in Figure 1.2.

3. *Although the industry is moving toward component-based assembly, most software continues to be custom built*
   - A software component should be designed and implemented so that it can be reused in many different programs.
   - Modern reusable components encapsulate both data and the processing applied to the data, enabling the software engineer to create new applications from reusable parts.
   - For example, today's graphical user interfaces are built using reusable components such as windows, pull-down menus, and a wide variety of interaction mechanisms.

2. Changing nature of software

- The nature of software has changed a lot over the years. It has changed from writing programs by individuals for their personal use to writing very complex software to run a nuclear power plant.
- Its nature mainly depends on the type of software used. Given below are the different types of software being used in different applications.

System Software

- System Software is a collection of programs written to provide service to other programs.
- It needs heavy interaction with computer hardware.
- It contains complex data structures and multiple external interfaces

  Examples: Compilers, Editors, File Management Utilities, other System Applications Drivers and Networking Software.

Application Software

- Application Software consists of standalone programs that are used to solve specific business needs.
- It is used to process technical data/technical decisions and control business functions in real time.

  Examples: Conventional Data Processing Applications, Real-Time Manufacturing Process Control, point-of-sale etc.

Engineering/Scientific Software

- It is characterized by conventional numerical algorithms.
- It is used to create interactive applications to take on real time.

  Examples: Computer Aided Design(CAD/CAM), System Simulation, Weather prediction system, Interactive Applications in Educational Field.

Embedded Software

- Software that resides within a product or system is called as Embedded Software.

  Examples: Keypad control for a Microwave Oven, Smart dustbins etc.

Product-line Software

- This type of software provides specific capability for use by many different customers.

  Examples: Word Processing, Spreadsheets, Computer Graphics, Database Management, Multimedia & Entertainment and Business Financial Applications.

Web Applications

- It can be considered as a set of linked hypertext files.
- Web Application Software has grown relevant as E-Commerce & B2B applications grow in importance.

  Examples: E-commerce sites, Air line reservation system, IRCTC etc.

Artificial Intelligence Software

- This type of software uses Non-Numerical Algorithms to solve complex problems.

Examples: Robotics, Expert Systems, Pattern Recognition(image and voice), Artificial Neural Networks, Theorem Proving, Game Playing.

Open Source

- Open Source Software refers to the software whose source code is public to everyone to develop, test or improve.

  Examples: Linux Operating System, Apache Web Server Application, LibreOffice Application, GNU Image Manipulation Application.

3. Software myths

- Are the false beliefs that managers, customers, and developers have on the software development.

3.1 Management myths

Myth1: Development problems can be solved by developing and documenting standards.

Reality: Standards have been developed by companies and standards organizations. They can be very useful. However, they are frequently ignored by developers because they think that they are irrelevant and sometimes incomprehensible.

Myth2: Development problems can be solved by using state-of-the art tools.

Reality: Tools may help, but there is no magic. Problem solving requires more than tools. It requires great understanding.

Myth3: If we fall behind schedule in developing software, we can just put more people on it.

Reality: If software is late, adding more people will merely make the problem worse. This is because the people already working on the project need to educate the newcomers. So, this does not immediately reduce the work.

Myth4: If I decide to outsource the software project to a third party, I can just relax and let that firm build it.

Reality: If an organization does not understand how to manage and control software projects internally, it will invariably struggle when it outsources software projects.

3.2 Customer myths

Customers often underestimate the difficulty of developing software. Sometimes marketing people encourage customers in their misbeliefs.

Myth1: A general statement of objectives is sufficient to begin writing programs— we can fill in the details later.

Reality: A poor definition of the problem is the major cause of failed software. A detailed description of the information domain, functions, behavior, performance, interfaces, design constraints, and validation criteria is essential before writing the programs. These can be determined only after thorough communication between customer and developer.

Myth2: Project requirements continuously change, but change can be easily accommodated because software is flexible.

Reality: It is true that software requirements change, but the impact of change varies with the time at which it is introduced. If change is requested in design, then it costs 5 times more as if it is done in analysis. If it is requested in coding, then it costs 10 more, in testing it is 50 times more and if it is requested after delivery, then its cost increases enormously.

3.3 Developer's (Practitioner's) myths

Practitioner's often want to be artists, but the software development craft is becoming an engineering discipline. However myths remain:

Myth1: Once we write the program and get it to work, our job is done.

Reality: Commercially successful software may be used for decades. Developers must continually maintain such software: they add features and repair bugs. Maintenance costs predominate over all other costs; maintenance may be 60% of the development costs. This myth is true only for shelfware --- software that is never used.

Myth2: Until I get the program "running" I have no way of assessing its quality.

Reality: One of the most effective software quality assurance mechanisms is formal technical review. Software reviews can effectively detect the problems in requirements documents, design documents, test plans, and code.

Myth3: The only deliverable work product for a successful project is the working program.

Reality: A working program is only one part of a software delivery. Apart from this several other documents such as analysis, design and testing documents, user manuals etc. may also be created during software development.

Myth4: Software engineering will make us create voluminous and unnecessary documentation and will invariably slow us down.

Reality: Software engineering is not about creating documents. It is about creating quality. Better quality leads to reduced rework. And reduced rework results in faster delivery times.

4. The Software Problem

- The software developed by a student as part of his laboratory work consists of few hundred Lines of Code (LOC). Such systems tend to have a very limited purpose and a very short life span. We can afford to throw them away and replace them with an entirely new software rather than attempt to reuse them or repair them.
- On the other hand, industrial strength software is developed by a team of people and consists of few thousand Lines of Code (LOC). These are the applications that exhibit a very rich set of behaviors and are used by a large number of customers. Such systems tend to have a very long life span. Ex. Air-traffic control system, Railway Reservation System etc.
- Thus, the problem domain for software engineering is developing an industrial strength software. This should be produced at reasonable cost, in a reasonable time, and should be of good quality.

- Thus, the basic elements of industrial strength software are cost, schedule and quality.

4.1 Cost

- Cost of software is measured based on its size. The size of the software is measured in Lines of Code (LOC) or Thousand Lines of Code (KLOC).
- As the main cost of producing software is the manpower employed, the cost of developing software measured in terms of LOC (or) KLOC per person-months.
- Generally, software companies charge the client between $3000 - $15000 per person month.
- For Example, assume that size of software is 50 million LOC and a person can write 5000 LOC in a month. Let the company charging $6000 per person month.
- Then the cost of software can be calculated as

  Person months required = 500,00,000/5000 = 10000

  Cost = 10000X6000 = $6,00,00,000

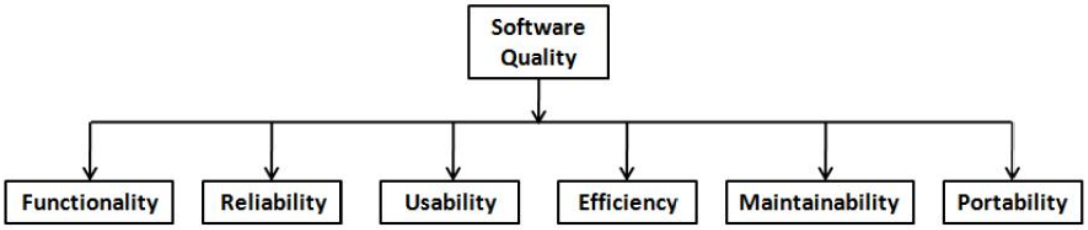
Schedule

- Schedule is very important for developing projects.
- Business trends are dictating that the time to market a product should be reduced. It means that software needs to be developed faster, and within the specified time.
- Unfortunately, the history of software shows that it is usually delivered late.
- Therefore, reducing the cost and time for software development are central goals of software engineering.

Quality

- Besides cost and schedule, the other major factor driving software engineering is quality.
- Today, quality is one of the main mantras, and business strategies are designed around it.

- The international standard on software quality suggests six main attributes of software quality as shown in the figure below.



Software quality attributes

These attributes can be defined as follows

- **Functionality:** The capability to provide functions which meet stated needs of software.
- **Reliability:** The capability to provide failure-free service.
- **Usability:** The capability to be understood, learned, and used.
- **Efficiency:** The capability to provide appropriate performance relative to the amount of resources used.
- **Maintainability:** The capability to be modified for purposes of making corrections, improvements, or adaptation.
- **Portability:** The capability to be adapted for different specified environments without applying actions.
- One measure of quality is the number of defects in the delivered software per unit size (generally taken to be thousands of lines of code, or KLOC).
- Current best practices in software engineering have been able to reduce the defect density to less than 1 defect per KLOC.

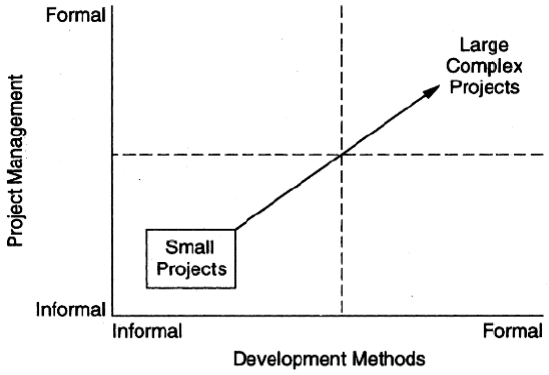
Though cost, schedule, and quality are the main driving forces of industry strength software, there are some other characteristics of it.

Scale

- Most industrial-strength software systems tend to be large and complex, requiring tens of thousands of lines of code. For example, sizes of some of the well-known software products are given below;

Software	Size (KLOC)
Windows 10	50,000
Google Chrome	7,000
Facebook	62,000
Ms Office	45,000
Google	2000,000
Human Genome	3300,000
Any Commercial Software (IRCTC, Banking s/w etc.)	50,000

- Developing a large system requires a different set of methods compared to developing a small system. i.e. the methods used for developing small systems often do not scale up to large systems.
- As an example, consider the problem of obtaining the opinion poll of people in a room as well as across the country. It is obvious that, the methods used for obtaining the opinion poll of people in a room will just not work for obtaining the opinion poll of people across the country. A different set of methods will have to be used for this and will require considerable management, execution and validation.
- Similarly, methods that one can use to develop programs of a few hundred lines of code cannot work for the software consisting of hundred thousand lines of code. A different set of methods must be used for developing such large software.
- Any software project involves the use of engineering and project management. In small projects, informal methods for development and management can be used. For large projects, more formal methods are used. This is shown below.



- As shown in the above figure, more formal methods are used for developing large and complex projects to make sure that cost, schedule, and quality are under control.

Change

- Change is another characteristic of the problem domain and should be handled properly.
- It is obvious that, all the requirements of the system are not known at the beginning. As the development proceeds and time passes, additional requirements are identified, which need to be incorporated in the software being developed. This requires that, suitable methods are to be developed to accommodate the change efficiently. Otherwise, change requests can trouble the project and can consume up to 30 to 40% of the development cost.

UNIT-I
Assignment-Cum-Tutorial Questions
SECTION-A

Objective Questions

1) What is Software? [ ]

a) Software is set of programs.

b) Software is documentation and configuration of data.

c) Both a and b

d) None of the mentioned

2) What are the characteristics of software? [ ]

a) Software is developed or engineered; it is not manufactured in the classical sense.

b) Software doesn't "wear out".

c) Software can be custom built or custom build.

d) All mentioned above

3) The process of developing a software product using software engineering principles and methods is referred to as, _____. [ ]

a) Software myths

b) Scientific Product

c) Software Evolution

d) None of the above

4) Software consists of _____. [ ]

a) Set of instructions + operating procedures

b) Programs + documentation + operating procedures

c) Programs + hardware manuals

d) Set of program

5) The extent to which the software can continue to operate correctly despite the introduction of invalid inputs is called as [ ]

a) Reliability

b) Robustness

c) Fault Tolerance

d) Portability

e) All of the above.

6) As per an IBM report, “31%of the project get cancelled before they are completed, 53% overrun their cost estimates by an average of 189% and for every 100 projects, there are 94 restarts”. What is the reason for these statistics? [ ]

a)Lack of adequate training in software engineering

b)Lack of software ethics and understanding

c)Management issues in the company

d) All of the mentioned

7) Compilers, Editors software comes under which type of software? [ ]

a) System software b) Application software

c) Scientific software d) None of the above

8) Which of the following cannot be applied with the software according to Software Engineering Layers? [ ]

a) Process b) Methods

c) Manufacturing d) None of the above.

9) Choose the correct option according to the given statement. [ ]

Statement 1: Software is a physical rather than a logical system element.
Statement 2: Computer software is the product that software engineers design and build.

Statement 3: Software is a logical rather than a physical system element.
Statement 4: Software is a set of application programs that are built by software engineers.

a) Statement 1 and 2 are correct.

b) Only Statements 2 and 3 are correct.

c) Statements 2, 3 and 4 are correct

10)From the following which quality deals with maintaining the quality of the software product? [ ]

a) Quality assurance b)Quality control

b) Quality efficiency d)None of the above

11)Which one of the following is not a symptom of the present software crisis: [ ]

a) Software is expensive

b) It takes too long to build a software product

c) Software is delivered late

d) Software products are required to perform very complex tasks

12) Which one of the following characteristics of software products being developed is not a symptom of software crisis? [ ]

a) Fail to meet user requirements b) Expensive

c) Highly interactive d) Difficult to alter, debug, and enhance

13) Why is writing easily modifiable code important? [ ]

a) Easily modifiable code results in quicker run time

b) Most real world programs require change at some point of time or other

c) Most text editors make it mandatory to write modifiable code

d) Several developers may write different parts of a large program

SECTION-B

SUBJECTIVE QUESTIONS

1) Define Software and Software Engineering? List out the important characteristics of software.
2) Discuss the changing nature of the software.
3) Identify different Myths and Realities related to software. Explain briefly.
4) Describe the major driving forces of a Software Project.
5) Illustrate different Software Quality Attributes? Explain briefly.
6) Give a conclusion about the statement “Software is easy to change, because Software is flexible”
7) Analyze how the Failure Curve of Hardware and Software can be differentiated?
8) Categorize some problems that will come up if the methods you currently use for developing small software are used for developing large software systems.
9) Suppose a program for solving a problem cost C and industrial strength software for solving that problem costs 10 C. where do you think this extra 9 C cost is spent? suggest a possible breakdown of this extra cost.

SECTION-C

GATE QUESTIONS

1) If you are given extra time to improve the reliability of the final product developing a software product, where would you spend this extra time?