SOFTWARE ENGINEERING

Subject Code: 10IS51 **I.A. Marks : 25**
Hours/Week : 04 **Exam Hours: 03**
Total Hours : 52 **Exam Marks: 100**

PART – A

UNIT – 1 **6 Hours**
Overview: Introduction: FAQ's about software engineering, Professional and ethical responsibility.
Socio-Technical systems: Emergent system properties; Systems engineering; Organizations, people and computer systems; Legacy systems.

UNIT – 2 **6 Hours**
Critical Systems, Software Processes: Critical Systems: A simple safety critical system; System dependability; Availability and reliability.
Software Processes: Models, Process iteration, Process activities; The Rational Unified Process; Computer Aided Software Engineering.

UNIT – 3 **7 Hours**
Requirements: Software Requirements: Functional and Non-functional requirements; User requirements; System requirements; Interface specification; The software requirements document.
Requirements Engineering Processes: Feasibility studies; Requirements elicitation and analysis; Requirements validation; Requirements management.

UNIT – 4 **7 Hours**
System models, Project Management: System Models: Context models; Behavioral models; Data models; Object models; Structured methods.
Project Management: Management activities; Project planning; Project scheduling; Risk management

PART – B

UNIT – 5 **7 Hours**
Software Design: Architectural Design: Architectural design decisions; System organization; Modular decomposition styles; Control styles. 33
Object-Oriented design: Objects and Object Classes; An Object-Oriented design process; Design evolution.

UNIT – 6 **6 Hours**
Development: Rapid Software Development: Agile methods;Extreme programming; Rapid application development.
Software Evolution: Program evolution dynamics; Software maintenance; Evolution processes; Legacy system evolution.

UNIT – 7 **7 Hours**
Verification and Validation: Verification and Validation: Planning; Software inspections; Automated static analysis; Verification and formal methods.
Software testing: System testing; Component testing; Test case design; Test automation.

UNIT – 8 **6 Hours**
Management: Managing People: Selecting staff; Motivating people; Managing people; The People Capability Maturity Model.
Software Cost Estimation: Productivity; Estimation techniques; Algorithmic cost modeling, Project duration and staffing.

Text Book:
1. Ian Sommerville: Software Engineering, 8th Edition, Pearson Education, 2007.
(Chapters-: 1, 2, 3, 4, 5, 6, 7, 8, 11, 14, 17, 21, 22, 23, 25, 26)

Reference Books:
1. Roger.S.Pressman: Software Engineering-A Practitioners approach, 7th Edition, Tata McGraw Hill, 2007.
2. Pankaj Jalote: An Integrated Approach to Software Engineering, Wiley India, 2009.

TABLE OF CONTENTS

UNIT -1 OVERVIEW

The economies of ALL developed nations are dependent on software. More and more systems are software controlled.

Software engineering is concerned with theories, methods and tools for professional software development.

FAQs About software engineering:

What is software?
Software is set of Computer programs associated with documentation & configuration data that is needed to make these programs operate correctly. A software system consists of a number of programs, configuration files (used to set up programs), system documentation (describes the structure of the system) and user documentation (explains how to use system).
Software products may be developed for a particular customer or may be developed for a general market.
Software products may be
• Generic - developed to be sold to a range of different customers
• Bespoke (custom) - developed for a single customer according to their specification

What is software engineering?

- Software engineering is an engineering discipline which is concerned with all aspects of software production.
- Software engineers should adopt a systematic and organized approach to their work and use appropriate tools and techniques depending on the problem to be solved, the development constraints and the resources available.

What is the difference between software engineering and computer science?

- Computer science is concerned with theory and fundamentals; software engineering is concerned with the practicalities of developing and delivering useful software
- Computer science theories are currently insufficient to act as a complete underpinning for software engineering

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 process
- System engineers are involved in system specification, architectural design, integration and deployment

What is a software process?
A set of activities whose goal is the development or evolution of software
Generic activities in all software processes are:
• Specification - what the system should do and its development constraints
• Development - production of the software system

• Validation - checking that the software is what the customer wants
• Evolution - changing the software in response to changing demands

What is a software process model?
A simplified representation of a software process, presented from a specific perspective
Examples of process perspectives are
• Workflow perspective - sequence of activities
• Data-flow perspective - information flow
• Role/action perspective - who does what

Generic process models
• Waterfall
• Evolutionary development
• Formal transformation
• Integration from reusable components

Socio-Technical Systems:

- A system is a purposeful collection of inter-related components working together towards some common objective.

- A system may include software, mechanical, electrical and electronic hardware and be operated by people.

- System components are dependent on other system components
  The properties and behavior of system components are inextricably inter-mingled

Problems of systems engineering

- Large systems are usually designed to solve 'wicked' problems
- Systems engineering requires a great deal of co-ordination across disciplines
  • Almost infinite possibilities for design trade-offs across components
  • Mutual distrust and lack of understanding across engineering disciplines
- Systems must be designed to last many years in a changing environment

Software and systems engineering

The proportion of software in systems is increasing. Software-driven general purpose electronics is replacing special-purpose systems
Problems of systems engineering are similar to problems of software engineering

Software is seen as a problem in systems engineering. Many large system projects have been delayed because of software problems.

Emergent properties

- Properties of the system as a whole rather than properties that can be derived from the
  the
  properties of components of a system

- Emergent properties are a consequence of the relationships between system components. They can therefore only be assessed and measured once the components have been integrated into a system.

Examples of emergent properties

1. *The overall weight of the system*
• This is an example of an emergent property that can be computed from individual component properties.
2. *The reliability of the system*
• This depends on the reliability of system components and the relationships between the components.
3. *The usability of a system*
• This is a complex property which is not simply dependent on the system hardware and software but also depends on the system operators and the environment where it is used.

Types of emergent property

1. Functional properties
• These appear when all the parts of a system work together to achieve some objective. For example, a bicycle has the functional property of being a transportation device once it has been assembled from its components.

2. Non-functional emergent properties
• Examples are reliability, performance, safety, and security.
These relate to the behaviour of the system in its operational environment. They are often critical for computer-based systems as failure to achieve some minimal defined level in
these properties may make the system unusable.

Because of component inter-dependencies, faults can be propagated through the system
System failures often occur because of unforeseen inter-relationships between Components It is probably impossible to anticipate all possible component relationships
Software reliability measures may give a false picture of the system reliability

System reliability engineering

1. *Hardware reliability*
• What is the probability of a hardware component failing and how long does it take to repair that component?
2. *Software reliability*
• How likely is it that a software component will produce an incorrect output. Software failure is usually distinct from hardware failure in that software does not wear out.
3. *Operator reliability*
• How likely is it that the operator of a system will make an error?
Influences on reliability

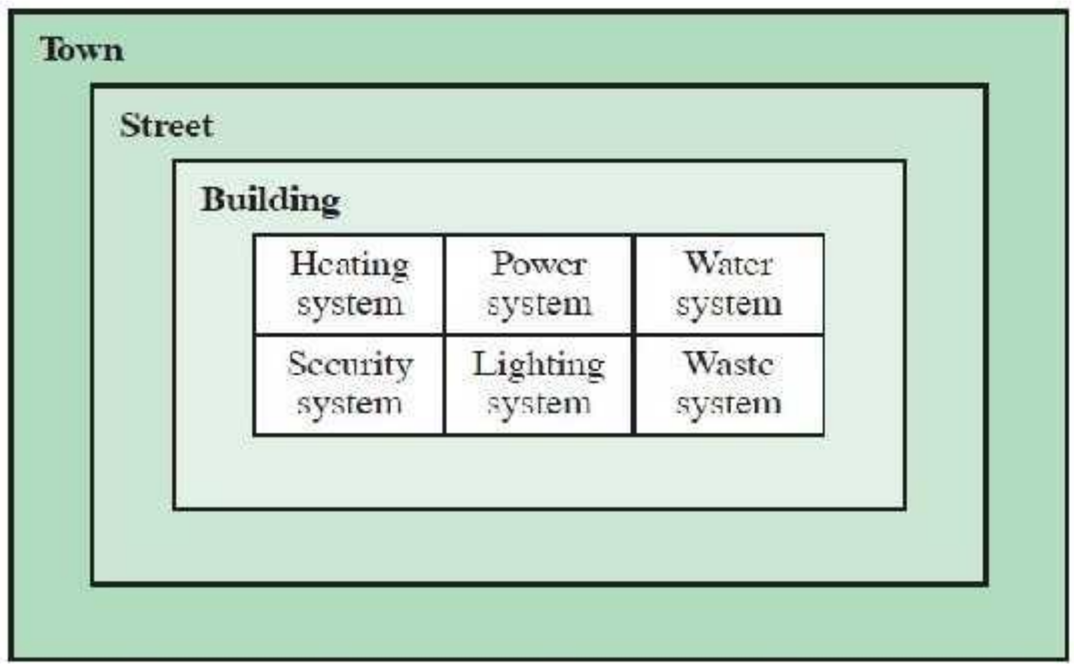
Reliability relationships

1. Hardware failure can generate spurious signals that are outside the range of inputs expected by the software
2. Software errors can cause alarms to be activated which cause operator stress and lead to operator errors

3. The environment in which a system is installed can affect its reliability

Systems and their environment

Systems are not independent but exist in an environment
System's function may be to change its environment. Environment affects the functioning of the system e.g. system may require electrical supply from its environment
The organizational as well as the physical environment may be important



Human and organisational factors

Process changes
• Does the system require changes to the work processes in the environment?
Job changes
• Does the system de-skill the users in an environment or cause them to change the way they work?
Organisational changes
• Does the system change the political power structure in an organisation?

System architecture modelling

An architectural model presents an abstract view of the sub-systems making up a system

may include major information flows between sub-systems

❚ Usually presented as a block diagram

❚ May identify different types of functional component in the model

System evolution

Large systems have a long lifetime. They must evolve to meet changing requirements

Evolution is inherently costly

• Changes must be analysed from a technical and business perspective

• Sub-systems interact so unanticipated problems can arise

• There is rarely a rationale for original design decisions

• System structure is corrupted as changes are made to it

Existing systems which must be maintained are sometimes called legacy systems

The system engineering process

Usually follows a 'waterfall' model because of the need for parallel development of different parts of the system

• Little scope for iteration between phases because hardware

changes are very expensive. Software may have to compensate for hardware problems

Inevitably involves engineers from different disciplines who must work together

• Much scope for misunderstanding here. Different disciplines use a

different vocabulary and much negotiation is required. Engineers may have personal agendas to fulfill.