ITC1052

Dr Nalaka Lankasena
Faculty of Technology
University of Sri Jayewardenepura

Software Engineering

ITC1052

Dr Nalaka Lankasena
Faculty of Technology
University of Sri Jayewardenepura

Table of Content

- Syllabus outline
- Recommended texts
- Evaluation criteria

  - Software engineering history
  - What is software, and attributes of good software?
  - What is software engineering?
  - Software engineering Body of Knowledge
  - Challenges of software engineering
  - Software products

Aim of the Module

- To provide students **broad overview of Software Engineering** principles and techniques to lay foundation for **developing quality software products** in **cost effective** way

Learning outcomes

LO-1: Understand the concept of the software life cycle in software engineering

LO-2: Describe major types of software development processes and select an appropriate model for development

LO-3:Understand the principles of requirements engineering process

LO-4:Understand the content and the importance of SRS document.

LO-5: Model requirements with use case diagrams

Syllabus Outline

- Introduction to software engineering (SE)
- Process models
- Requirement engineering

Topics in Detail

Introduction to software engineering (SE)

Types of software, Characteristics of software, Role of software, What is software engineering, Challenges in SE

Process models

Generic view of processes models, Waterfall model, Incremental process models: Incremental model, **Evolutionary process models**: Prototyping, **Spiral model**; Component Based Development

Requirement engineering

Types of requirements; Identifying the stakeholders, **Elicitation requirements**: Requirement gathering, Quality function deployment, **User scenarios**; Developing use-cases, **Negotiation and validating requirements;** Requirements writing and Documentations

Evaluation Criteria

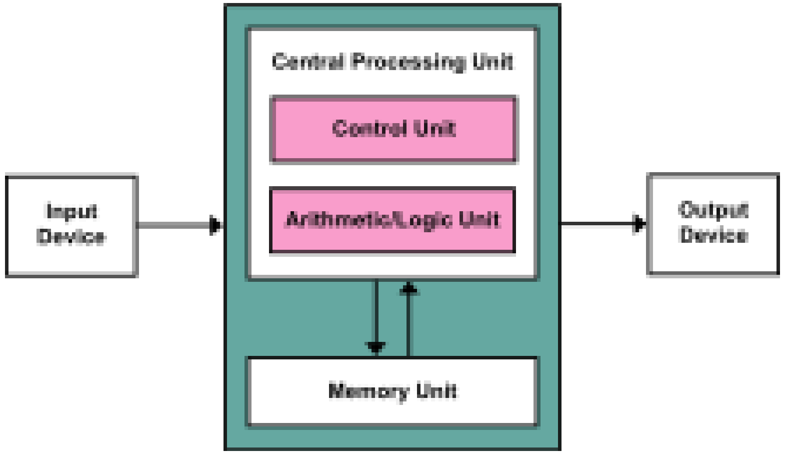
- CA - Continuous Assessments – 30%-40%
  - Classroom assignments/assessments
  - Projects/Reports/Case studies
  - Group activities and Presentations
  - Quizzes

- ESA - End Semester Examination – 70%-60%

Recommended Texts

- Sommerville, I., *Software Engineering*. 10th ed., Addison Wesley, 2017
- Pressman, R. *Software Engineering - A Practitioners Approach*. 9th ed., McGraw Hill, 2020

Software Engineering – History



- First digital computers appeared in the early 1940s, the instructions to make them operate were wired into the machine.

  It was realized that this design was not flexible and came up with the "**stored program architecture**“ (=stores program instructions in electronic memory) or von Neumann architecture.
- Division between "**hardware**" and "**software**" began with abstraction being used to deal with the complexity of computing.
- Programming languages started to appear in the 1950s Ex. Fortran, ALGOL, COBOL

… contd..

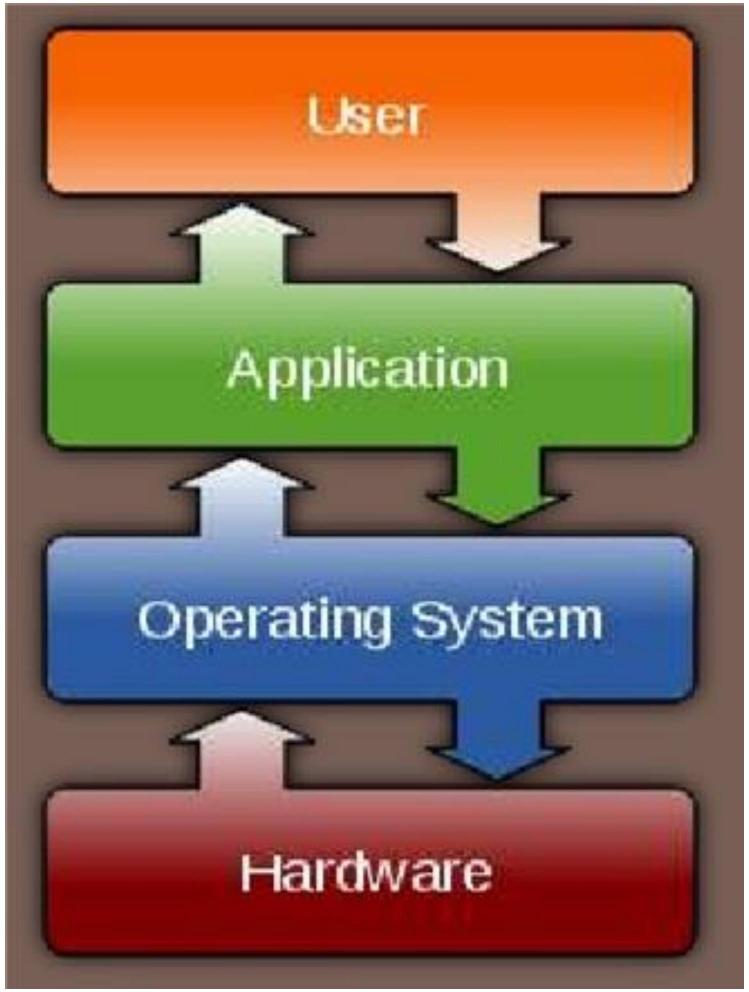
.. Contd.. Software Engineering – History

- The term '**software engineering**' was suggested at conferences organized by NATO in 1968 and 1969 to discuss the **'software crisis'**.
- The **software crisis** was the name given to the **difficulties encountered in developing large, complex systems** in the 1960s.
- It was proposed that the **adoption of an engineering approach** to software development would **reduce the costs** of software development and lead to **more reliable software**.
- The term "**software engineering**", was used first by Anthony Oettinger in 1967 and then used by Margaret Hamilton in 1968 as a title for the world's first conference on software engineering, sponsored and facilitated by NATO.

https://ifs.host.cs.st-andrews.ac.uk/Books/SE9/Web/History/
https://malkio.github.io/tribute-margaret-hamilton/

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**.

Why software is important ?

- The economies of almost all nations are dependent on software

- Software engineering expenditure represents a significant fraction of the Gross National Product (GNP) of developed countries.

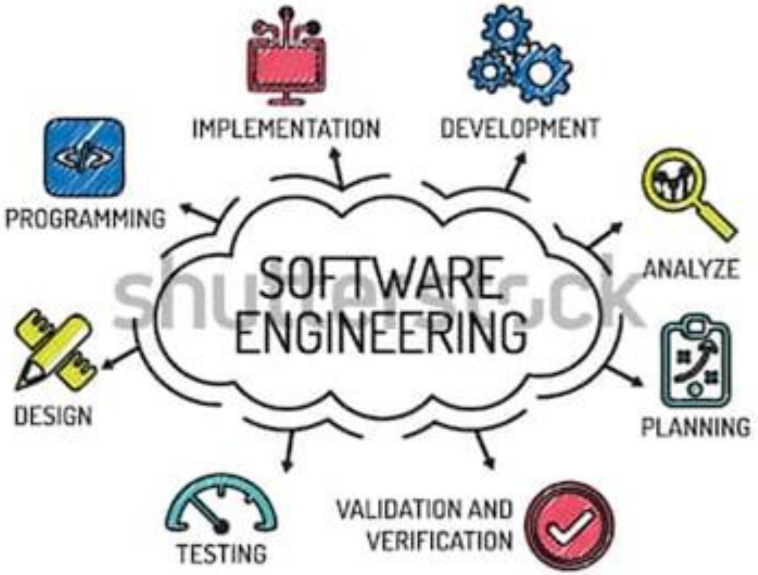
- More and more systems are software controlled: it permeates almost every aspect of our lives.

  (Automation, information processing, communication, Innovation, flexibility and adaptability, connectivity, user experience, research and development, economic growth, education)

- eGovernance (eGov) has also emerged as a tool for delivering government services, to citizens in a convenient, efficient and transparent manner.

Software Engineering

Software engineering is an **engineering discipline** that is concerned with **all aspects of software production** from the **early stages of system specification through to maintaining** the system after it has gone into use.



Engineering discipline

Engineers make things work using appropriate theories and methods to solve problems bearing in mind organizational and financial constraints.

Engineering is about getting results of the required quality within the schedule and budget

- **All aspects of software production**

  Not just technical process of development. Also project management and the development of tools, methods etc. to support software production.

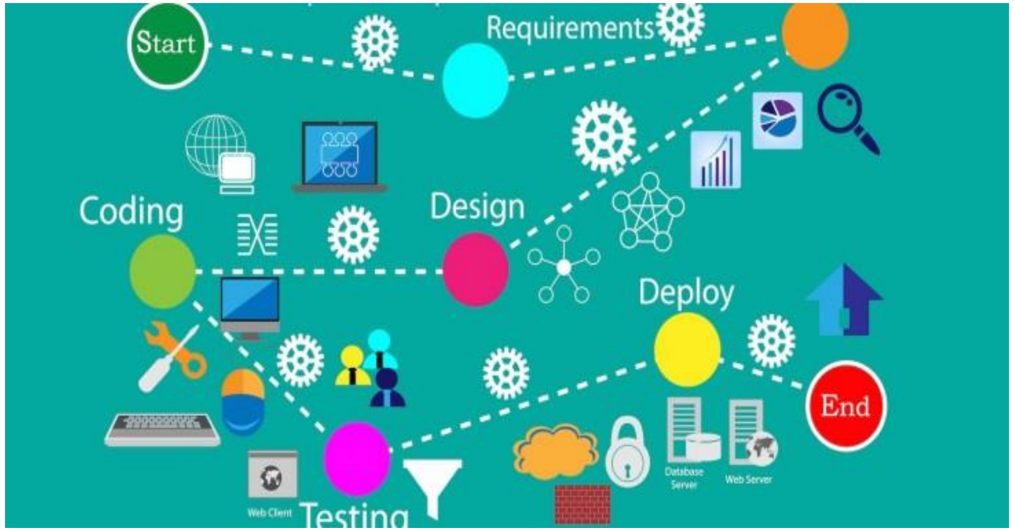
Importance of software engineering

- More and more, individuals and society rely on advanced software systems. We should produce **reliable and trustworthy** systems **economically and quickly**.

- It is usually **cheaper**, in the long run, to use software engineering methods and techniques for software systems rather than just write the programs as if it was a personal programming project.
  - For most types of systems, the **majority of costs are the costs of changing the software** after it has gone into use.

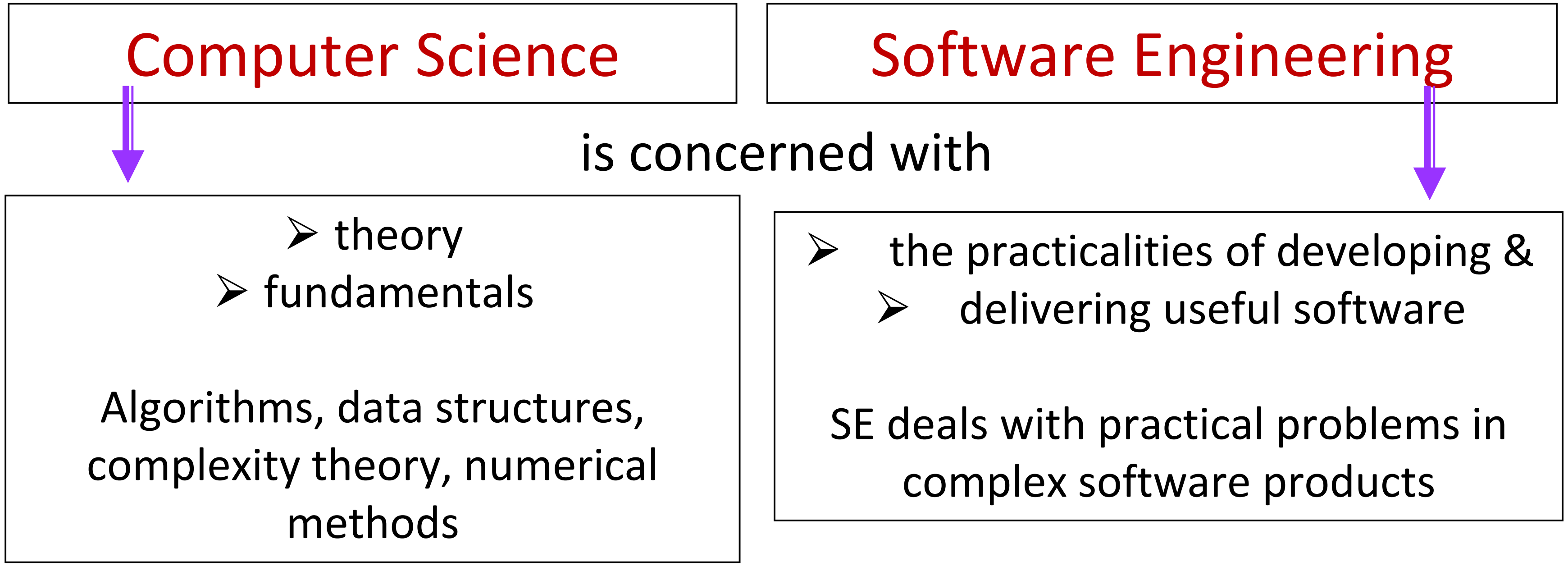
Difference between software engineering and computer science

Computer science focuses on **theory and fundamentals**; software engineering is concerned with the **practicalities of developing and delivering useful software and its maintenance**.



Contd…

Contd… **Difference between software engineering and computer science?**



Computer Science

Software Engineering

is concerned with

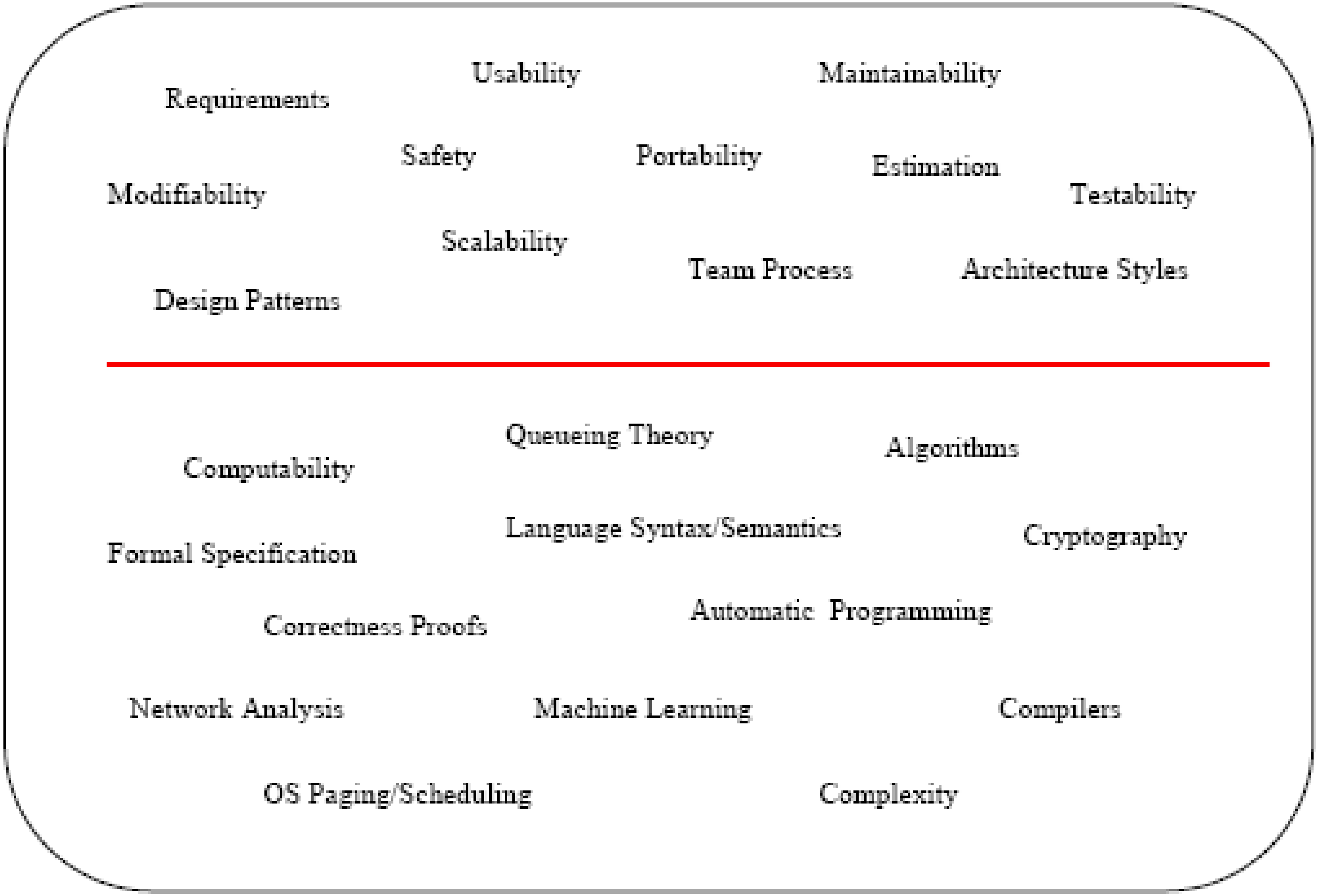
- ➢ theory
- ➢ fundamentals

Algorithms, data structures, complexity theory, numerical methods

- ➢ the practicalities of developing &
- ➢ delivering useful software

SE deals with practical problems in complex software products

Computer science theories are currently insufficient to provide complete support for software engineering, but it is a foundation for practical aspects of software engineering



Usability Maintainability

Requirements

Safety Portability Estimation

Modifiability Testability

Scalability

Team Process Architecture Styles

Design Patterns

Queueing Theory Algorithms

Computability

Language Syntax/Semantics Cryptography

Formal Specification

Automatic Programming

Correctness Proofs

Network Analysis Machine Learning Compilers

OS Paging/Scheduling Complexity

What are the key challenges in software engineering?

- Coping with **increasing diversity**, demands for **reduced delivery times** and developing **trustworthy software**.

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**.
  - Ex. **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.

Software products

- **Generic products**
  - Stand-alone systems that are marketed and **sold to any customer who wishes to buy them**.
    - Examples – PC software such as graphics programs, project management tools; CAD software; software for specific markets such as appointments systems for dentists.
  - The specification of what the software should do is owned by the software developer and decisions on software change are made by the developer.

- **Customized products**
  - Software that is commissioned by a specific customer to meet their own needs.
    - Examples – air traffic control software, traffic monitoring systems.
  - The specification of what the software should do is owned by the customer for the software and they make decisions on software changes that are required

Q&A
Questions
Answers