Module 1: Introduction to Software Engineering

Software Engineering

Faculty of Computing
Universiti Teknologi Malaysia

Outline

- Software engineering definition
- Software engineering as a layered technology
- Types of software
- Inherent difficulties in software engineering
- Software engineering quality focus
- Documentation standard

Note: The overall contents of the slide are based on the main reference that is Sommerville (2016) with other references specified directly in respective slides

Objectives

The objectives of this module are:

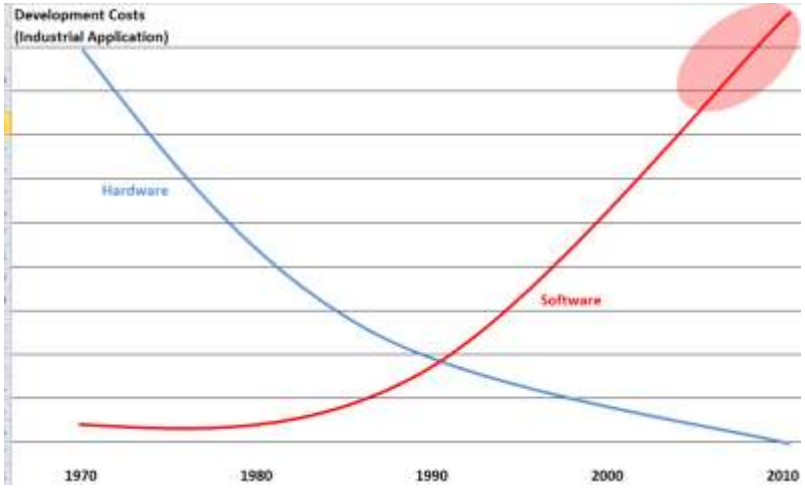
- To understand the importance of software engineering in software development
- To understand the definition of software engineering
- To know the difference between Software Engineering and Computer Science, Software Engineering and Software Programming
- To understand the importance of software quality and documentation standard

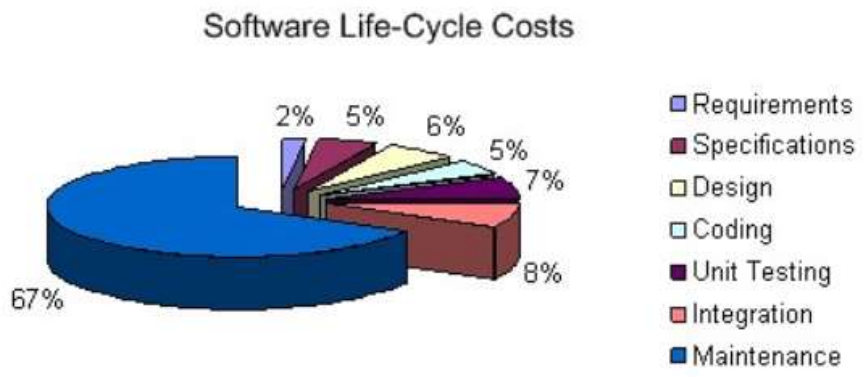
The Importance of Software Engineering

- Software engineering is important because software is needed in almost every industry, business and transactions
- If an application fails; a fast, efficient, and effective fix needs to happen as soon as possible

Software Engineering Costs

- Software costs are often greater than that of hardware
- Software costs more to maintain than it does to develop
- For systems with a long life, **maintenance costs** may be several times higher than development costs
- Therefore software engineering is concerned with **cost-effective** software development





Ref: Schach 2002 p.12

Image source: http://cerescontrols.com/tutorials-3/

Software Engineering in Software Development

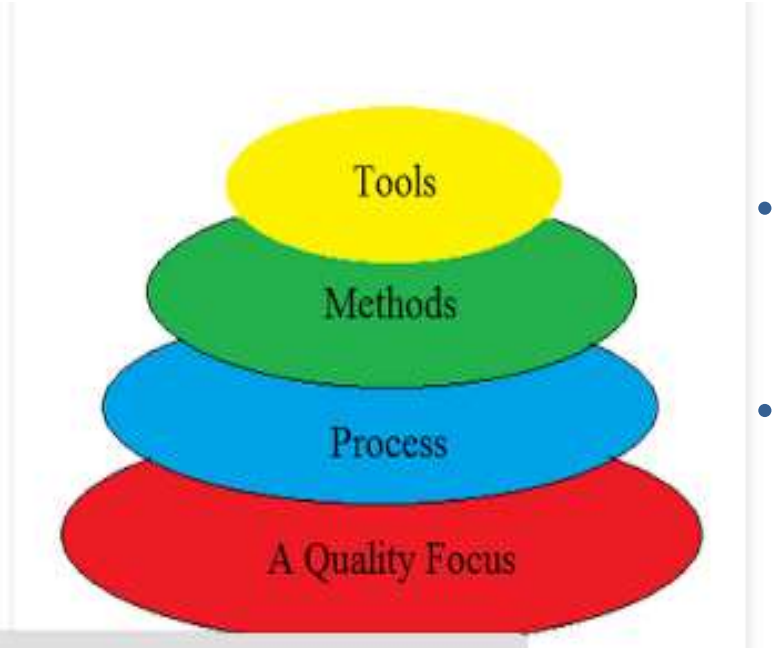
- Software engineering is concerned with **theories**, **methods** and **tools** for professional software development
- Software engineers should adopt a **systematic** and **organised approach** to their work and use appropriate tools and techniques depending on the problem to be solved, the development constraints and the resources available

Software Engineering Definition

- Software engineering is an **engineering discipline** that is concerned with all aspects of **software production**.

- "The application of a systematic, disciplines, quantifiable approach to the development, operation, and maintenance of software; that is the application of engineering to software." (IEEE, 1993)

Software Engineering as a Layered Technology



Note: The layered technology will be the focus in this lecture and throughout the semester

- **Quality Focus:** The bedrock that supports software engineering is quality focus where it ensures continuous process improvement culture.
- **Process:**Foundation for software engineering which enables rational and timely development of computer software
- **Methods:** provide technical how to's for building software. Involve different tasks including requirements analysis, design, program construction, testing and support. Methods also include modeling activities

Tools: provide automated or semi-automated support for the process and methods

Source:

Agarwal, U. (2019). Software Engineering, Kataria and Sons.

https://www.geeksforgeeks.org/layered-technology-in-software-engineering/

Software Products and Product Specification

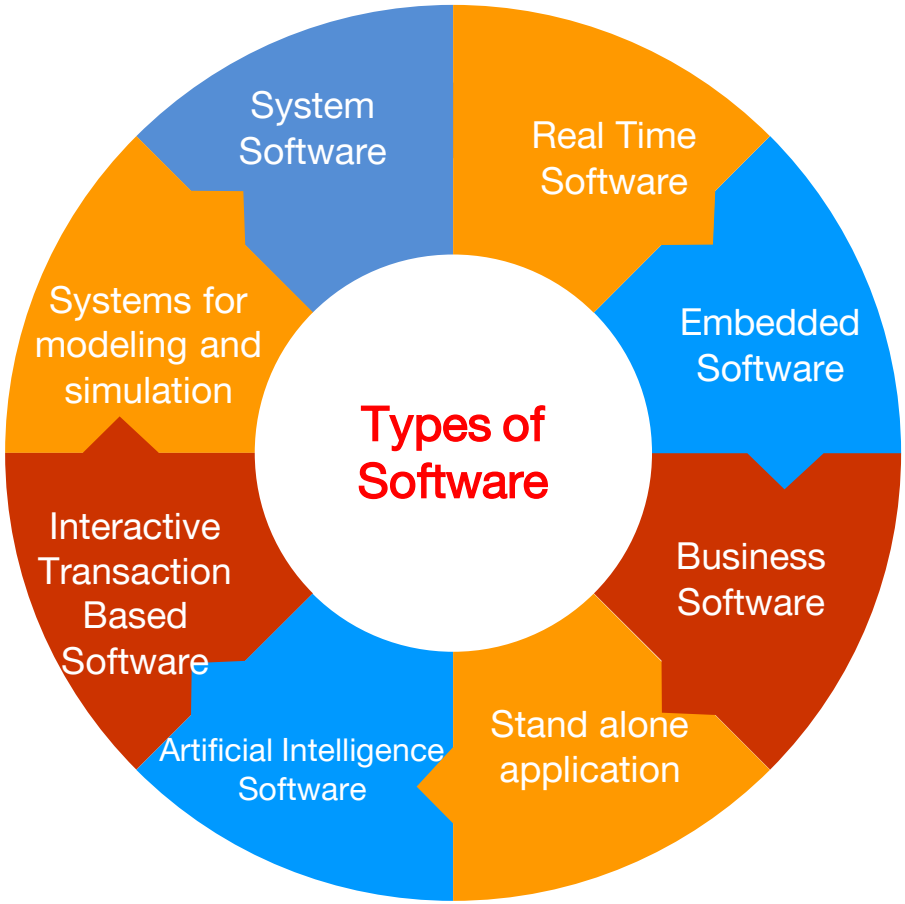
Generic Products

- 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 – embedded control systems, 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.

Software Engineering Diversity: Types of Software



Sources: Agarwal, U. (2019). Software Engineering, Kataria and Sons.
Sommerville, I. (2016). Software Engineering 10th edition, Pearson.

Different Types of Software…

- **System software**
  - To manage resources and provide clean interfaces to OS services
- **Real-time software**
  - monitors, analyzes, and controls real-world events as they occur in real-time; design efforts for this type of software is to guarantee the tasks complete within deadlines
  - Response time typically ranges from 1 millisecond to 1 second
- **Business software**
  - Management information system (MIS) that accesses one or more databases containing business information (e.g. payroll, inventory)
- **Embedded software**
  - Runs on specific devices such as microwave ovens, washing machines, automobiles
  - software resides in read only memory and used to control products and systems for the consumer and industrial market

Different Types of Software

- **System for modeling and simulation**

  These are systems developed by scientist and engineers to model physical processes or situations, which often computationally intensive and require high-performance parallel systems for execution

- **Stand alone application**

  Word processing, spreadsheets, computer graphics, multimedia, entertainment, personal and business financial applications, etc.

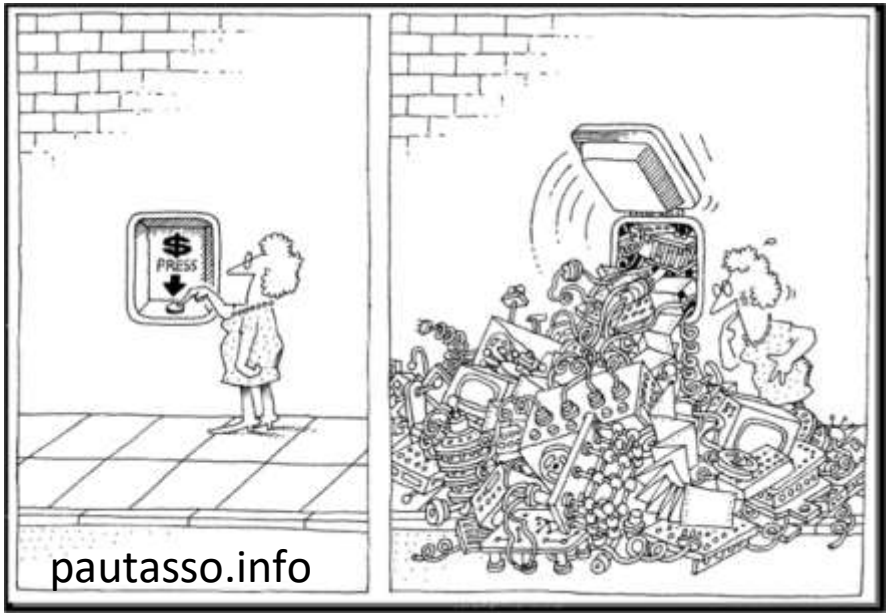
- **Interactive transaction-based software**

  Applications that execute on a remote computer/server and accessed by users from their own devices that include web applications such as e-commerce that nowadays can be accessed as mobile applications

- **Artificial intelligence software**
  - Uses non-numerical algorithms to solve complex problems
  - Applications: robotics, expert systems, pattern recognition, adaptive control, game playing

Inherent Difficulties

- Software inherent difficulties include:
  - Software complexity
  - Software changeability



Software Complexity…

Why is it complex?

No one really understands a large software product as a whole

- faults in specifications are made simply because of a lack of understanding of all aspects of the product
- no matter how trivial, the various pieces of the product will interact
- difficulty due to invisible nature of relationships and imperfect model of reality

No silver bullet

- There are many different types of software system and there is no universal set of software techniques that is applicable to all of these
- The software engineering methods and tools used depend on the type of application being developed, the requirements of the customer and the background of the development team

Software Complexity…

The complexity comes from:

- **Application domain**
  - The problems are often very **complex**
  - The developers are **not domain experts**
- **Communication among stakeholders (clients, developers)**
  - The stakeholders use **different vocabulary**:
    - domain experts <=> developers <=> developers
  - Human languages are inherently **ambiguous**
  - The stakeholders have **different background knowledge**
- **Management of large software development projects**
  - Need to divide the project into pieces and reassemble the pieces
  - Need to coordinate many people

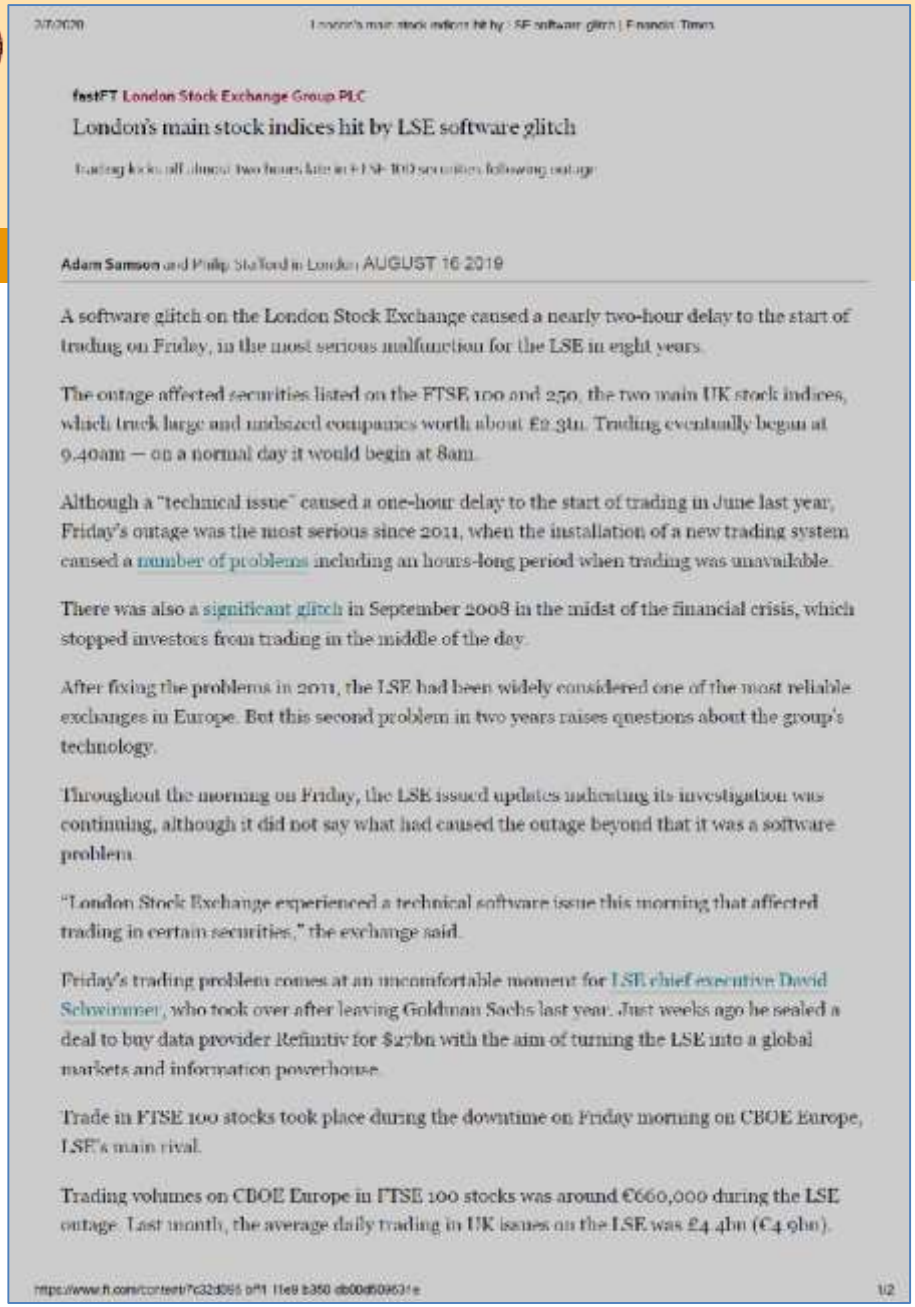
Software Complexity…

- Software complexity leads to software quality problems:
  - Unreliable e.g. ARIANE 5 rocket
  - Unsafe e.g. London Ambulance
  - Abandoned e.g. London Stock Exchange
  - Inflexible hard to change/maintain
- Software project management problems
- Often over schedule and over budget by an order of magnitude
- Software engineer productivity problems

Software Complexity…

- **Ariane 5 rocket:**
  - Its maiden flight on June 4, 1996 ended in the launcher being exploded because of a chain of software failures
- **London Ambulance:**
  - Because of a succession of software engineering failures the system was introduced that failed twice in the autumn of 1992
  - Although the monetary costed at “only” about £9m, was small by comparison with other examples, it is believed that PEOPLE DIED due to the ambulance do not come as promptly as they would
- **London Stock Exchange (LSE):**
  - The estimated loss to customers was around £450m and the damage to the reputation of the LSE was incalculable

LSE Example



2/7/2020 London's main stock indices hit by LSE software glitch | Financial Times

FastFT London Stock Exchange Group PLC

London's main stock indices hit by LSE software glitch

Trading locks up almost two hours late in FTSE 100 securities following outage

Adam Samson and Philip Stafford in London AUGUST 16 2019

A software glitch on the London Stock Exchange caused a nearly two-hour delay to the start of trading on Friday, in the most serious malfunction for the LSE in eight years.

The outage affected securities listed on the FTSE 100 and 250, the two main UK stock indices, which track large and midsized companies worth about £2.3tn. Trading eventually began at 9.40am — on a normal day it would begin at 8am.

Although a "technical issue" caused a one-hour delay to the start of trading in June last year, Friday's outage was the most serious since 2011, when the installation of a new trading system caused a number of problems including an hours-long period when trading was unavailable.

There was also a significant glitch in September 2008 in the midst of the financial crisis, which stopped investors from trading in the middle of the day.

After fixing the problems in 2011, the LSE had been widely considered one of the most reliable exchanges in Europe. But this second problem in two years raises questions about the group's technology.

Throughout the morning on Friday, the LSE issued updates indicating its investigation was continuing, although it did not say what had caused the outage beyond that it was a software problem.

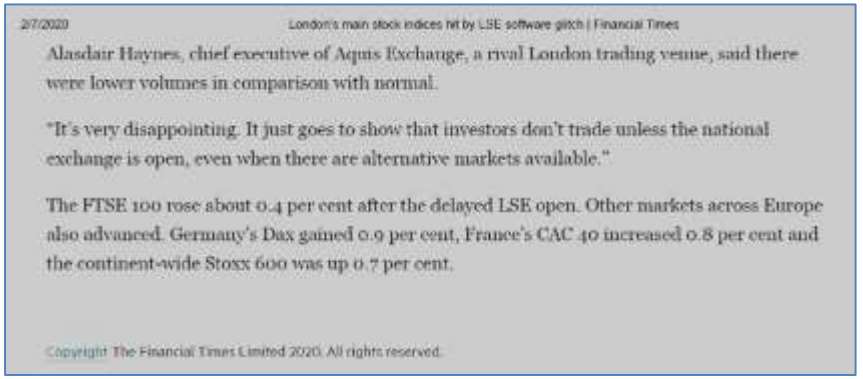
"London Stock Exchange experienced a technical software issue this morning that affected trading in certain securities," the exchange said.

Friday's trading problem comes at an uncomfortable moment for LSE chief executive David Schwimmer, who took over after leaving Goldman Sachs last year. Just weeks ago he sealed a deal to buy data provider Refinitiv for $27bn with the aim of turning the LSE into a global markets and information powerhouse.

Trade in FTSE 100 stocks took place during the downtime on Friday morning on CBOE Europe, LSE's main rival.

Trading volumes on CBOE Europe in FTSE 100 stocks was around €660,000 during the LSE outage. Last month, the average daily trading in UK issues on the LSE was £4.4bn (€4.9bn).

https://www.ft.com/content/7c32d086-bff4-11e9-b350-db00d509634e 1/2



2/7/2020 London's main stock indices hit by LSE software glitch | Financial Times

Alasdair Haynes, chief executive of Aquis Exchange, a rival London trading venue, said there were lower volumes in comparison with normal.

"It's very disappointing. It just goes to show that investors don't trade unless the national exchange is open, even when there are alternative markets available."

The FTSE 100 rose about 0.4 per cent after the delayed LSE open. Other markets across Europe also advanced. Germany's Dax gained 0.9 per cent, France's CAC 40 increased 0.8 per cent and the continent-wide Stoxx 600 was up 0.7 per cent.

Source: Financial Times
https://www.ft.com/content/7c32d086-bff4-11e9-b350-db00d509634e

Software Changeability

- Major changes to software will always and frequently be demanded
  - successful software is required to provide more functionality
  - software survives hardware
- Changing software ("maintenance")
  - is difficult due to dynamic nature of relationships between its interacting parts

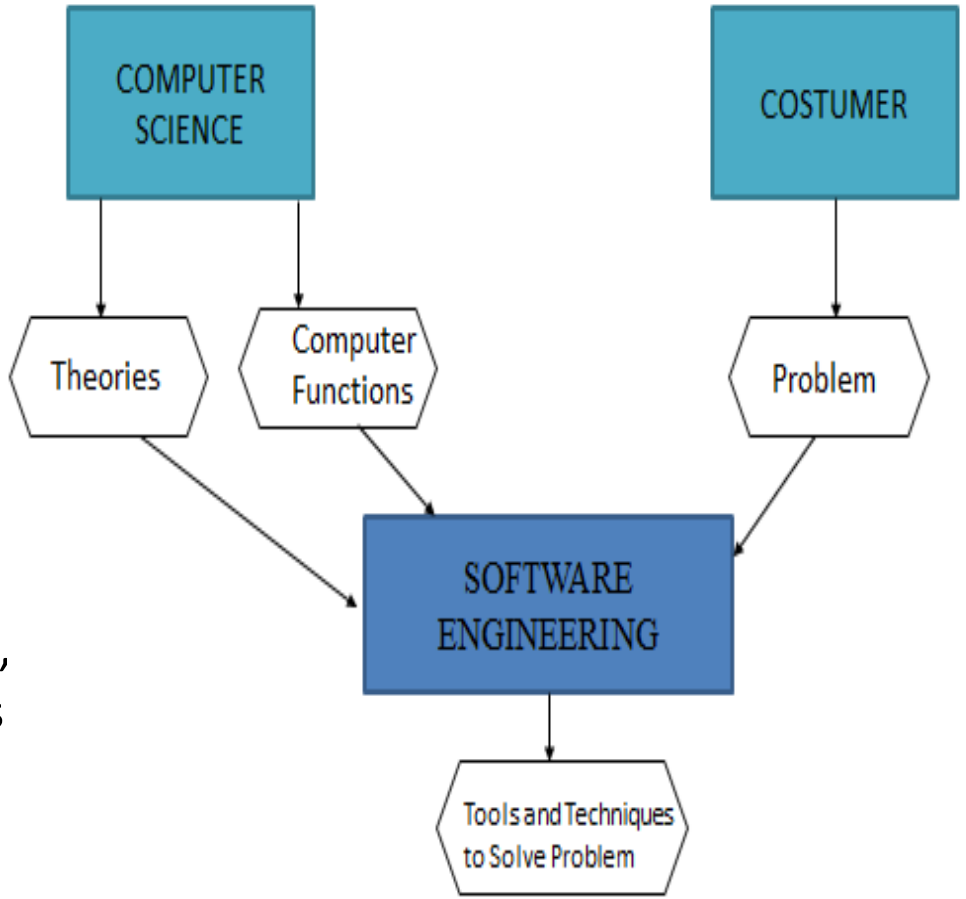
Computer Science vs. Software Engineering

Computer science

- The study of computation & information processing, both in hardware and software
- The discipline concerned with developing large applications

Software engineering

- Covers not only the technical aspects of building software systems, but also management issues, such as directing programming teams, scheduling, and budgeting



Source : Agarwal, U. (2012). Software Engineering, Kataria and Sons.

Software Engineering vs. Software Programming

Software Programming	Software Engineering
Single developer	Teams of developers with multiple roles
“Toy” applications	Complex systems
Short lifespan	Indefinite lifespan
One-of-a-kind systems	System families
Built from scratch	May be reused to reduce costs
Minimal maintenance	Maintenance accounts for over 60% of overall development costs

Quality Focus

- External qualities are visible to the user
  - reliability, efficiency, usability
- Internal qualities are the concern of developers
  - they help developers achieve external qualities
  - verifiability, maintainability, extensibility, evolvability, adaptability

Product

Process

Product vs. Process Qualities:

- Product qualities concern the developed artifacts
  - maintainability, understandability, performance
- Process qualities deal with the development activity
- Products are developed through process
  - maintainability, productivity, timeliness

Quality Focus: Example of Software Quality…

Product characteristic	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.
Verifiability	Its properties can be verified easily performed by formal analysis or testing internal quality
Efficiency	Software should not make wasteful use of system resources such as memory and processor cycles. Efficiency therefore includes **responsiveness**, processing time, memory utilisation, 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.

Quality Focus: Example of Software Quality

Product characteristic	Description
Usability	The higher the usability of software, the easier it is for users to work with it. There are several aspects of usability, including **learnability** for novices, **efficiency** of use for experts and **handling of errors**.
Reliability	Software is more reliable if it has **fewer failures**. Since software engineers do not deliberately plan for their software to fail, reliability depends on the number and types of mistakes they make.
Reusability	A software component is reusable if it can be used in several different systems with **little or no modification**. High reusability can reduce the long term costs faced by the development team.
Correctness	**Ideal quality**: established with respect to the requirements specification **Absolute software quality**: any deviation from the requirements makes the software incorrect, regardless of how minor or serious is the consequence of the deviation
Robustness	"Reasonable" behavior in unforeseen circumstances; subjective; a specified requirement is an issue of correctness; an unspecified requirement is an issue of robustness

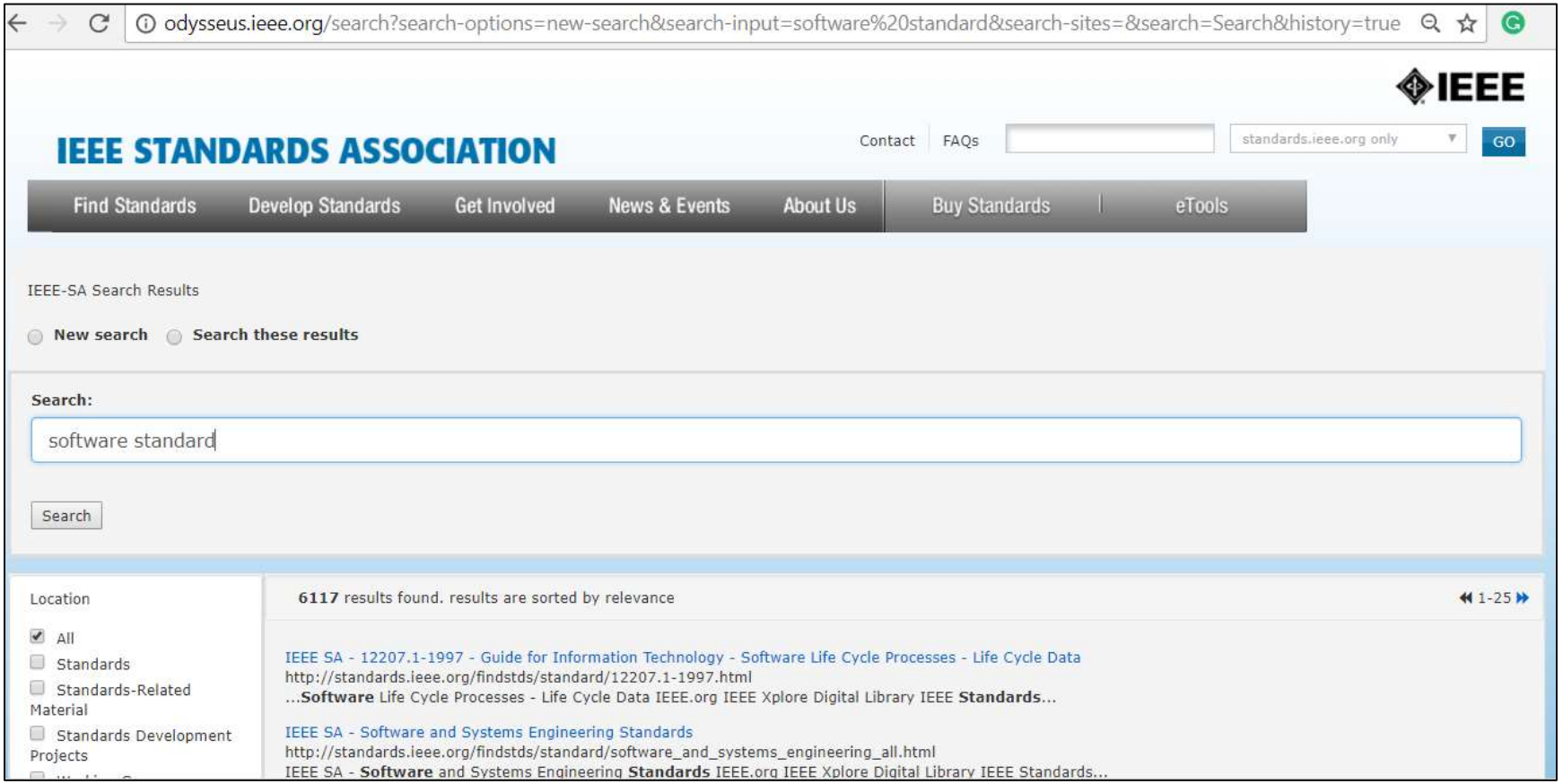
Quality Priority

- All the quality are important. However the importance of each quality differs from different application area and it can be costly where improving one quality can be an expense in cost to other quality.
- **Depends on the application area**:-
  - Information systems (library cataloguing, personnel system)
  - Safety critical system (Nuclear power plant)
  - Distributed systems (banking system)
  - Embedded systems (vending machine, patient monitoring system)
- **Influenced by cost**
  - Efficiency vs. Cost: Improving efficiency may make the design less easy to be understood which make it costly to be maintained.
  - Reliability vs. Cost: High reliability entails repeatedly checking for errors, adding redundant computations which make efficiency cannot be achieved.

Software Standard and Document

- Standard contributes to software quality mainly in producing documents for each Software Development Life Cycle (SDLC)
- Example of document standards by IEEE:
  - IEEE-Std830-1998-Software Requirements Specifications
  - IEEE-Std1016-2009-Software Design Desriptions
  - IEEE 829 - Standard for Test Documentation Overview-Test Plan Outline
  - IEEE Std829-2008-Software and System Test Documentation

Example: IEEE Standards



The expectation

The student is able to deliver:

a) Proposal

b) Software Requirements Specification

c) Software Design Document

d) Software Testing Document