Topic 1: Introduction to Software Engineering

Software Engineering

Faculty of Computing
Universiti Teknologi Malaysia

Topic Outline

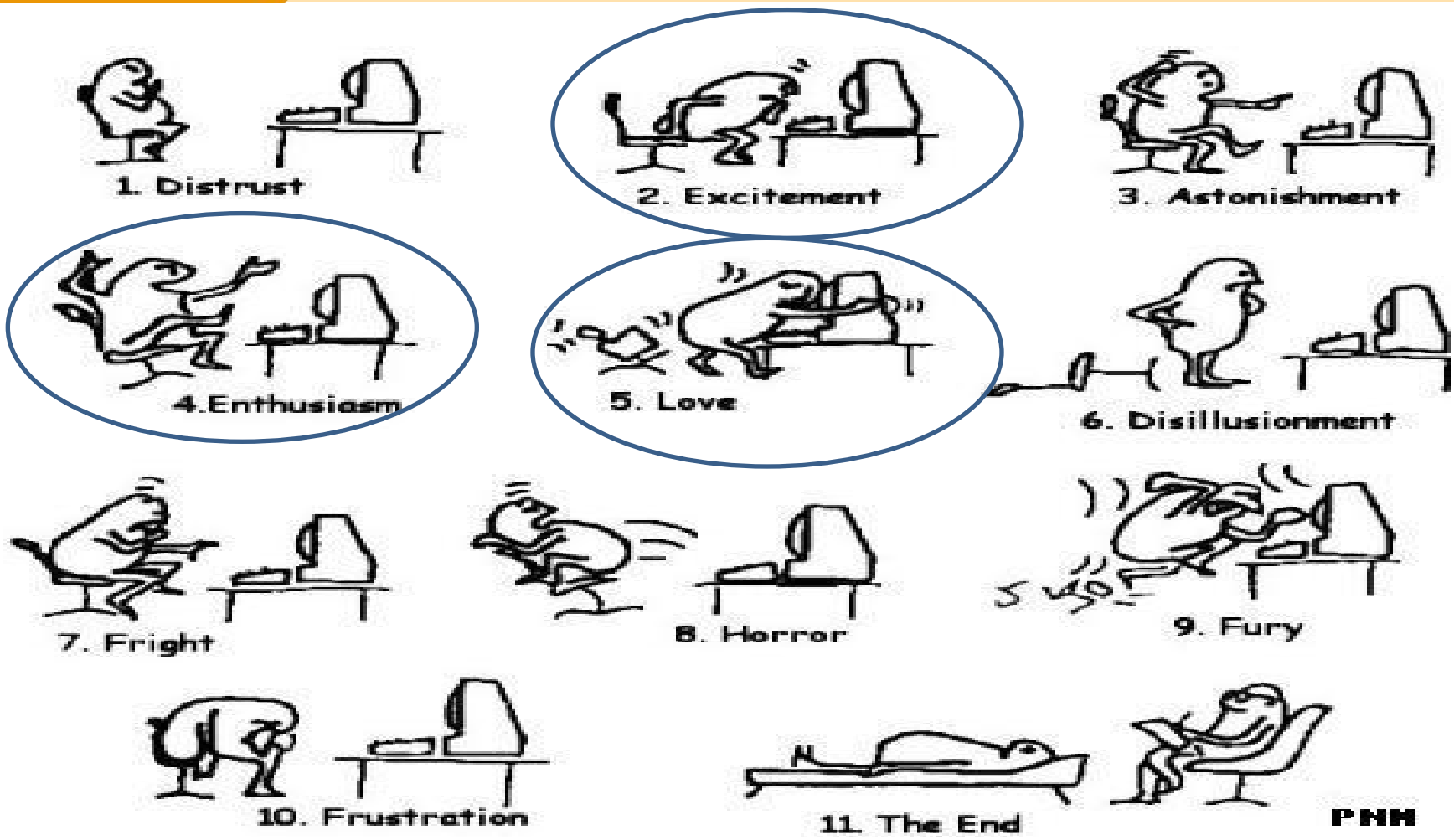
- Software engineering definition
- Types of software
- Inherent difficulties in software engineering
- Software engineering quality focus
- Software standard

Objectives

The objectives of this topic are :

- To understand the definition of Software Engineering.
- To know the difference between Software Engineering and Computer Science; Software Engineering and Software Programming.
- Understand what software engineering is and why it is important in software development.
- To understand the importance of Software Quality.

Software Engineering -> Engineers



Software Engineering

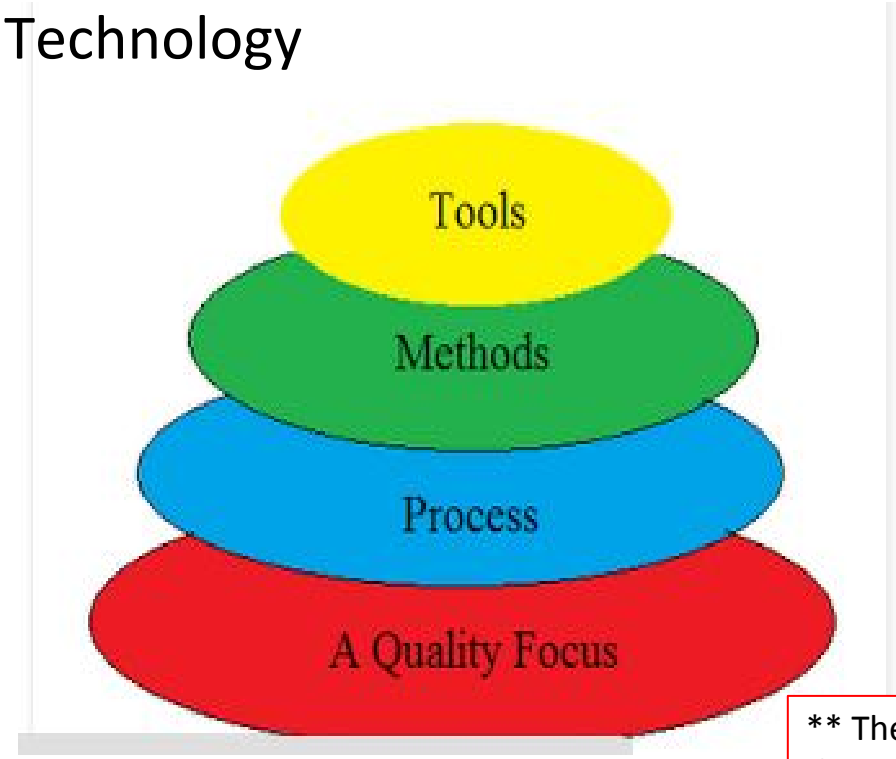
- The economies of ALL developed nations are dependent on software – on-line tax, on-line banking etc.
- More and more systems are software controlled
- Software engineering is concerned with **theories**, **methods** and **tools** for professional software development.

Software Costs

- Software costs often dominate computer system costs. The costs of software on a PC are often greater than the hardware cost.
- Software costs more to maintain than it does to develop. For systems with a long life, maintenance costs may be several times development costs.
- Software engineering is concerned with cost-effective software development.

Software Engineering as a Layered Technology

Software Engineering as Layered Technology



- **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

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

Source:

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

http://sesolution.blogspot.my/p/software-engineering-layered-technology.html

Software Products and Specifications

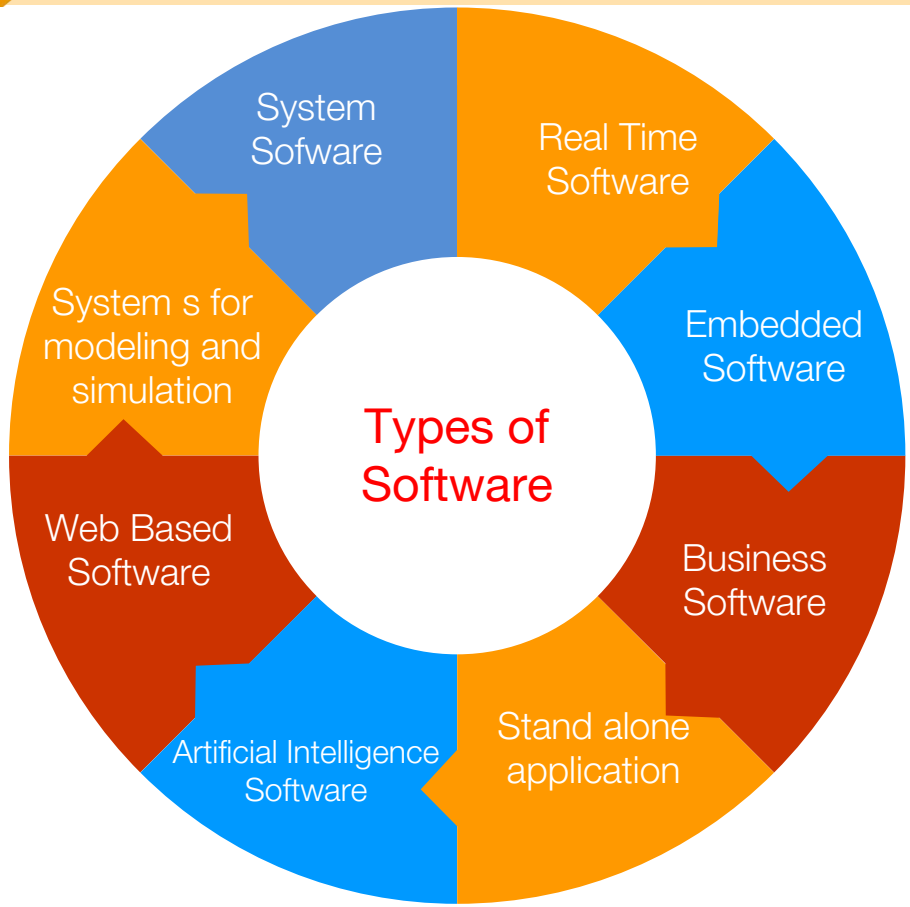
Generic Products

- 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



Source : Agarwal, U. (2012). Software Engineering, Kataria and Sons.
Sommerville, I. (2011) Software Engineering 9th edition, Pearson

Frequently Asked Questions About Software Engineering

Question	Answer
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, design, implementation, validation and 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.
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.

Frequently Asked Questions About Software Engineering

Question	Answer
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.
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.

Software Applications

- System software–collection of programs written to service other programs
  - Heavy interaction with computer hardware, multiple users, concurrent operation, resource sharing, sophisticated process management, complex data structures, multiple external interfaces
  - Examples: operating system components, drivers, telecommunications processors, compilers, editors, file management utilities

- Application software –standalone programs that solve specific business or technical need
  - Examples: data processing applications, point-of-sale transaction processing, real-time manufacturing process control

Software Applications

- Business software–business information processing
  - Management information system (MIS) that accesses one or more databases containing business information (e.g., payroll, inventory)
- Engineering and scientific software(e.g., numerical estimations, simulation, etc.)
- PC software–(word processing, spreadsheets, computer graphics, multimedia, entertainment, personal and business financial applications, etc.)
- Web-based software
  - A set of linked hypertext files that present information using text and graphics, e-commerce, B2B applications

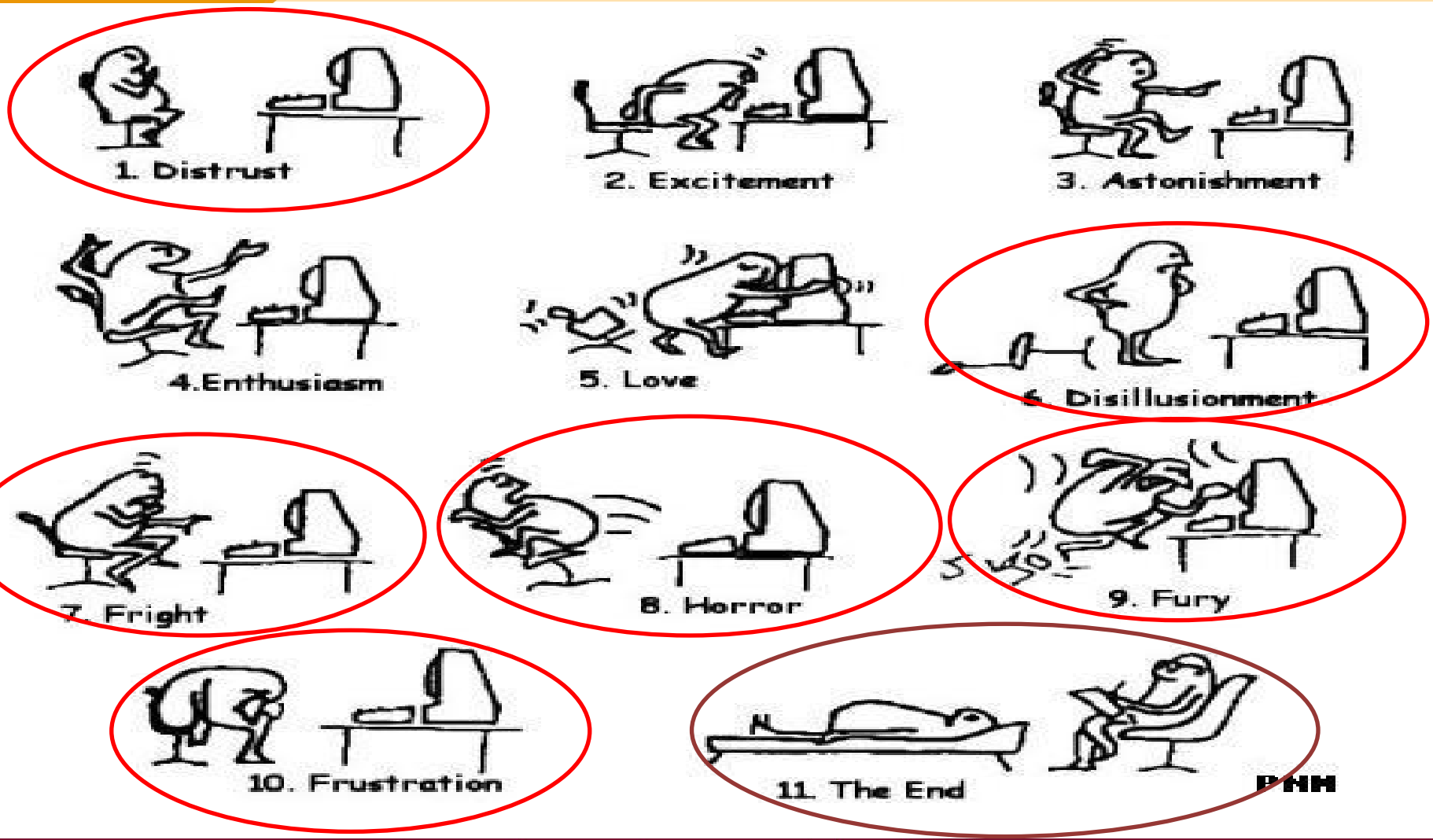
Software Applications

- Real-time software–monitors, analyzes, and controls real-world events as they occur in real-time
  - Response time typically ranges from 1 millisecond to 1 second
- Embedded software–control products and systems for consumer and industrial markets
- Artificial intelligence software –uses non-numerical algorithms to solve complex problems
  - Applications: robotics, expert systems, pattern recognition, adaptive control.
- Ubiquitous computing –growth of wireless networking, use of small mobile devices, laptops, etc.

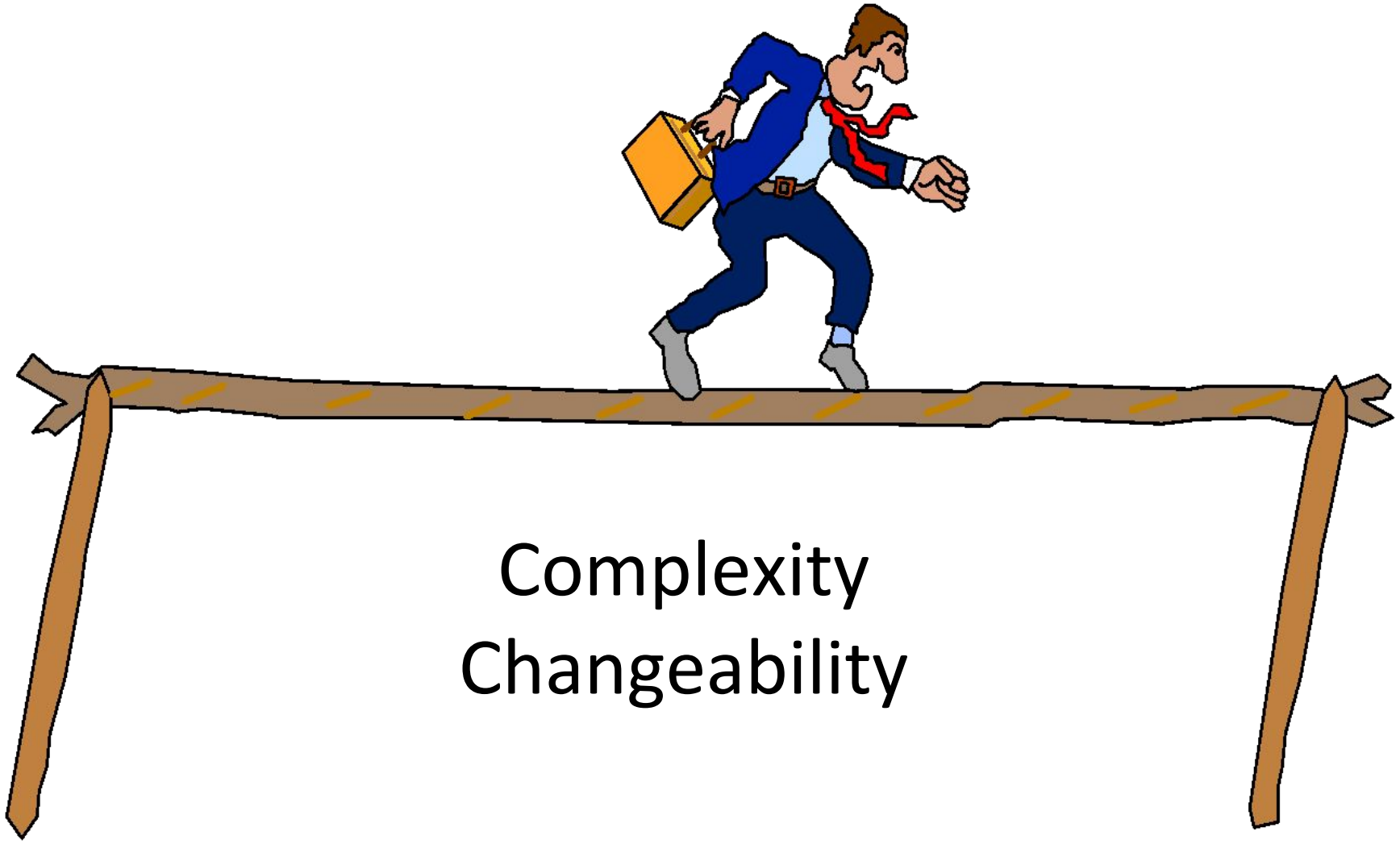
Software is …(Cont.)

- Important
  - pervasive and essential part of almost all organizations
  - key part of many products
- Big business
  - several hundred billion dollars/year spent worldwide and growing
- But **complex** to develop
  - Windows 95: 15MLOC
    (plus 5,000 estimated bugs!)
  - Windows XP: 40MLOC

Software Engineering -> Engineers



Inherent Difficulties



Complexity
Changeability

Complexity

- 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

Software Engineering Complexity

- 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 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 Leads to..

- Software quality problems
  - unreliable ➔ ARIANE 5 rocket
  - unsafe ➔ London Ambulance
  - Abandoned ➔ 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

For large software projects:
- 25% are canceled
- 50% take longer than planned
- 75% are operational failures

Software Complexity Leads to..

- **Ariane 5**
  - 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, especially defects in project management, a system was introduced that failed twice in the autumn of 1992. Although the monetary cost, at “only” about £9m, was small by comparison with other examples, it is believed that people died who would not have died if ambulances had reached them as promptly as they would have done without this software failure.
- **London Stock Exchange**
  - Taurus was a planned automated transaction settlement system. The project was canceled in 1993 after having lasted more than five years and costing around £75m; the estimated loss to customers was around £450m; and the damage to the reputation of the London Stock Exchange was incalculable.

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

What is Software Engineering

- Software engineering is an **engineering discipline** that is concerned with all aspects of **software production**.
- 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
IEEE Definition (1993)

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

Computer Science vs. Software Engineering

Computer science

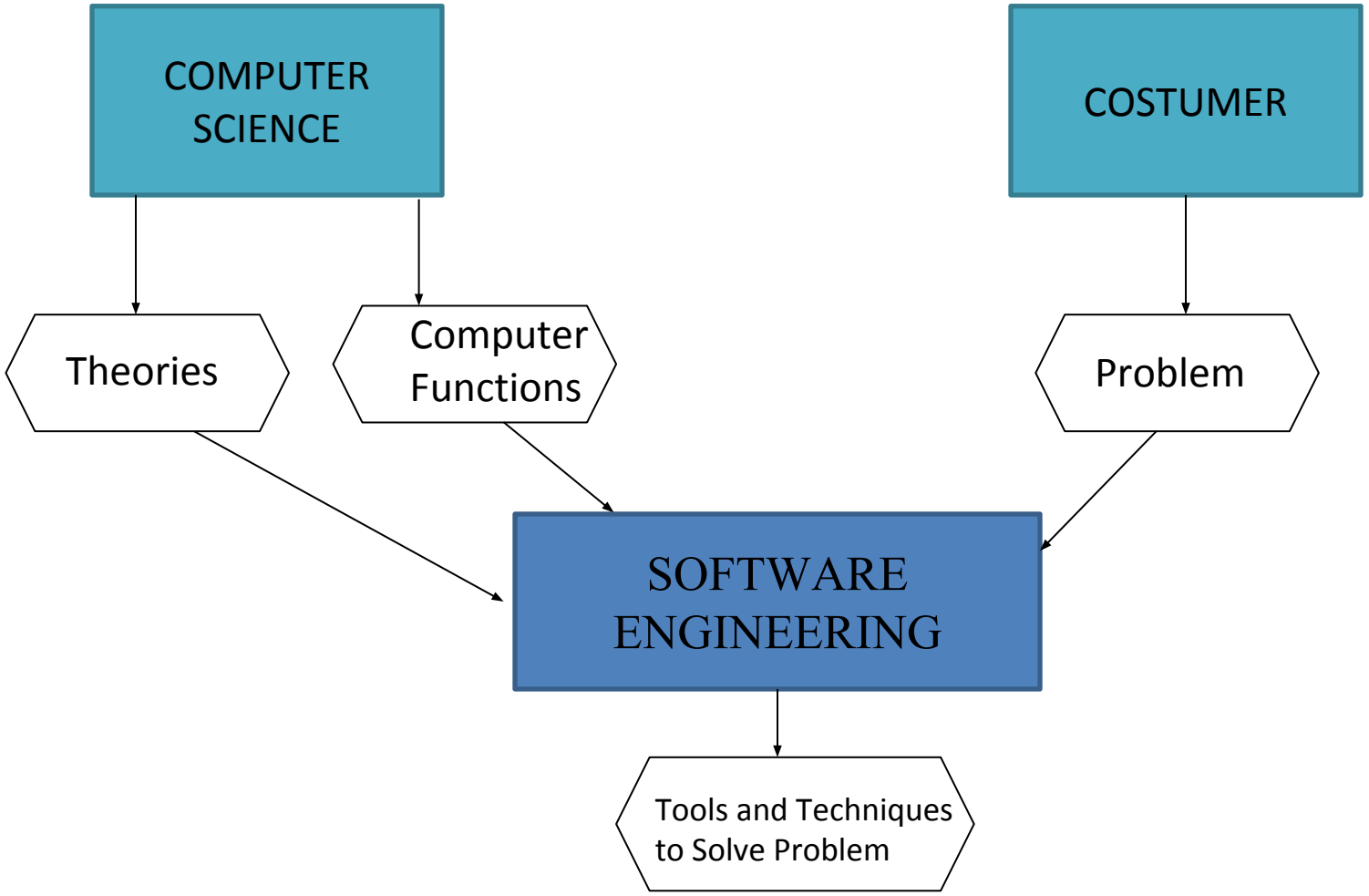
- The study of computation & information processing, both in hardware and software.
- The discipline concerned with developing large applications.

Software engineering

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

"A scientist builds in order to learn; an engineer learns in order to build."
— Fred Brooks

Where Does Software Engineering Fit in



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 reuse to reduce costs
Minimal maintenance	Maintenance accounts for over 60% of overall development costs

Quality Focus: Essential Attributes of Good Software

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.
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 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.

Classification of Software Quality

- 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 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

Examples of Software Quality

- **Correctness**
  - ideal quality
  - established with respect to the requirements specification
  - absolute

- **Reliability**
  - statistical property
  - probability that software will operate as expected over a given period of time
  - relative

E.g. ATM, Bulb

Examples of Software Quality (Cont…)

- **Robustness**
  - "reasonable" behavior in unforeseen circumstances
  - subjective
  - a specified requirement is an issue of correctness; an unspecified requirement is an issue of robustness

E.g. ATM Card (incorrect position)

- **Verifiability**
  - Its properties can be verified easily.
  - performed by formal analysis or testing
  - internal quality

- **Understandability**
  - ability of developers to easily understand produced artifacts
  - internal product quality

Quality Priority

- **Depends on the application area**:
  - Information systems (library cataloguing, personnel system)
  - Real-time systems (mouse click, robotic)
  - Distributed systems (banking system)
  - Embedded systems (vending machine, patient monitoring system)

- **Influenced by cost**
  - Efficiency vs. Cost
  - Reliability vs. Cost
  - Maintainability vs. Cost.

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