Introduction:
System Analysis and Design

Chapter 1

King Saud University
College of Computer and Information Sciences
Department of Computer Science

Dr. S. HAMMAMI

Course Objectives

✓To provide students with new ways of looking at information in the world in order to solve business problems

✓To introduce students to concepts and methods of SAD

✓To describe the systems development life cycle (SDLC)

✓To teach students effective methods for gathering essential information during system analysis

✓To teach students approaches to documenting and modeling of gathered information

✓To teach students effective methods for designing systems to solve problems effectively ***using appropriate methodology and technology***

Introduction

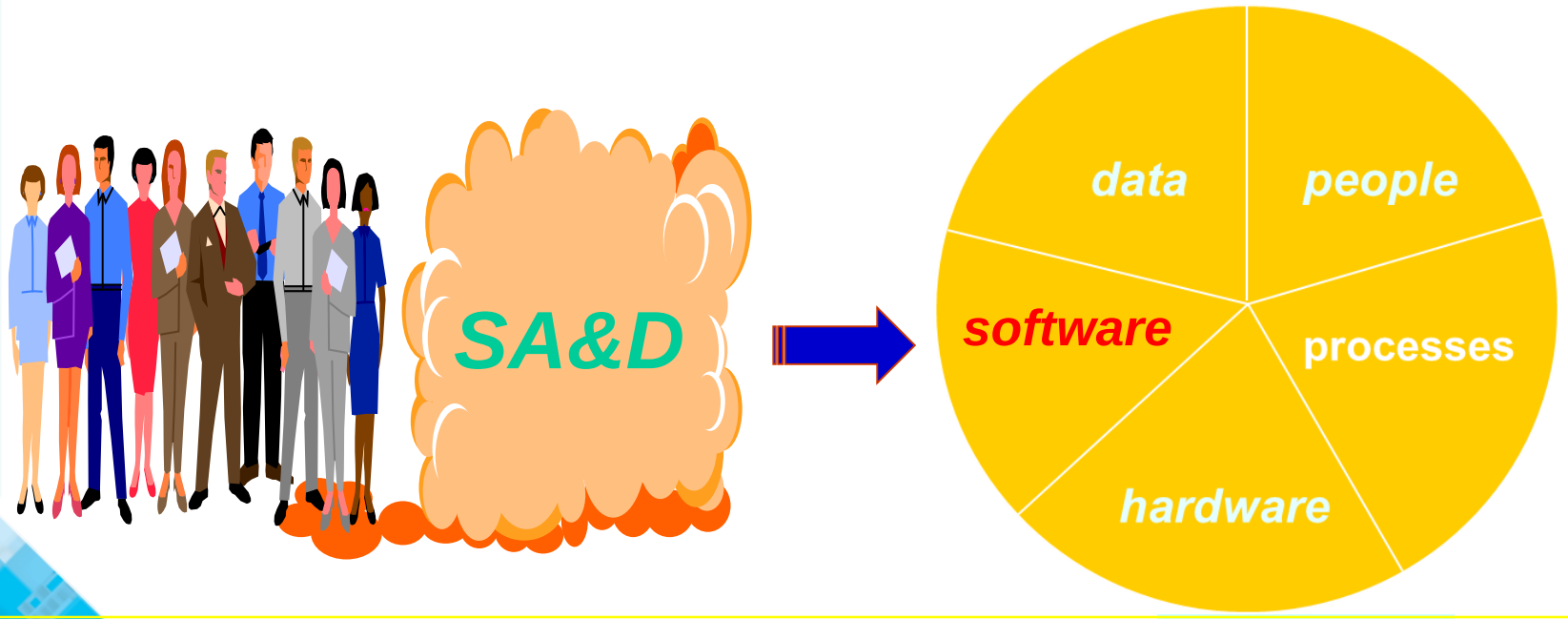
- **Systems Analysis means** understanding and specifying in detail ***what*** an information system should do.
- **System Design has to do with** specifying in detail ***how*** the parts of an information system should be implemented

Why is it important?

- Success of information systems depends on good SAD
- Widely used in industry - proven techniques
- Part of career growth in IT - lots of interesting and well-paying jobs!
- Increasing demand for systems analysis skills

Introduction

- *Systems Analysis and Design is the process people use to create information systems*



FAQs about software engineering

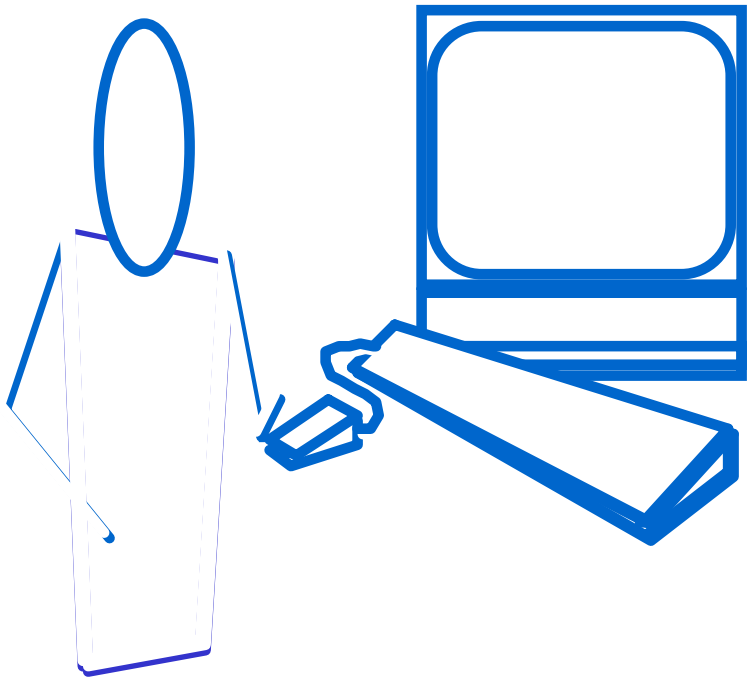
- What is software?
- What is software engineering?
- What is the difference between software engineering and computer science?
- What is the difference between software engineering and system engineering?
- What is a software process?
- What is **a** software process model**?**
- What are the costs of software engineering?
- What are software engineering methods?
- What is CASE (Computer-Aided Software Engineering)
- What are the attributes of good software?
- What are the key challenges facing software engineering?

What is Software?

Software is a set of items or objects that form a “configuration” that includes

- programs
- documents
- data ...

that is needed to make these programs operate correctly



The Nature of Software...

Software is abstract and intangible

- Hard to understand development effort

Software is easy to reproduce

- Cost is in its *development*

  —in other engineering products, manufacturing is the costly stage

The industry of the software is labor-intensive

- The process of the development is hard to automate

Types of Software...

Software products may be developed for a particular customer or may be developed for a general market:

Custom (Bespoke)

- For a specific customer according to their specification.
- Examples: air traffic control systems, ….

Generic

- Sold on open market
- Examples: databases, word processors, ….

Embedded

- Built into hardware
- Hard to change

Types of Software

Real time software

- E.g. control and monitoring systems
- Must react immediately
- Safety often a concern

Data processing software

- Used to run businesses
- Accuracy and security of data are key

Some software has both aspects

Variety of Software Products

Examples

Real time:	air traffic control
Embedded systems:	digital camera, GPS
Data processing:	telephone billing, pensions
Information systems:	web sites, digital libraries
Sensors:	weather data
System software:	operating systems, compilers
Communications:	routers, mobile telephones
Offices:	word processing, video conferences
Scientific:	simulations, weather forecasting
Graphical:	film making, design

etc., etc., etc., ...

Software engineering diversity

- 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

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.

What is Software Engineering?...

The process of solving customers' problems by the systematic development and evolution of large, high-quality software systems within cost, time and other constraints.

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**
  - Using appropriate theories and methods to solve problems bearing in mind organizational and financial constraints.
- ✧ **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 need to be able to 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 system, the majority of costs are the costs of changing the software after it has gone into use.

Software engineering fundamentals

- Some fundamental principles apply to all types of software system, irrespective of the development techniques used:
  - Systems should be developed using a managed and understood development process. Of course, different processes are used for different types of software.
  - Dependability and performance are important for all types of system
  - Understanding and managing the software specification and requirements (what the software should do) are important
  - Where appropriate, you should reuse software that has already been developed rather than write new software

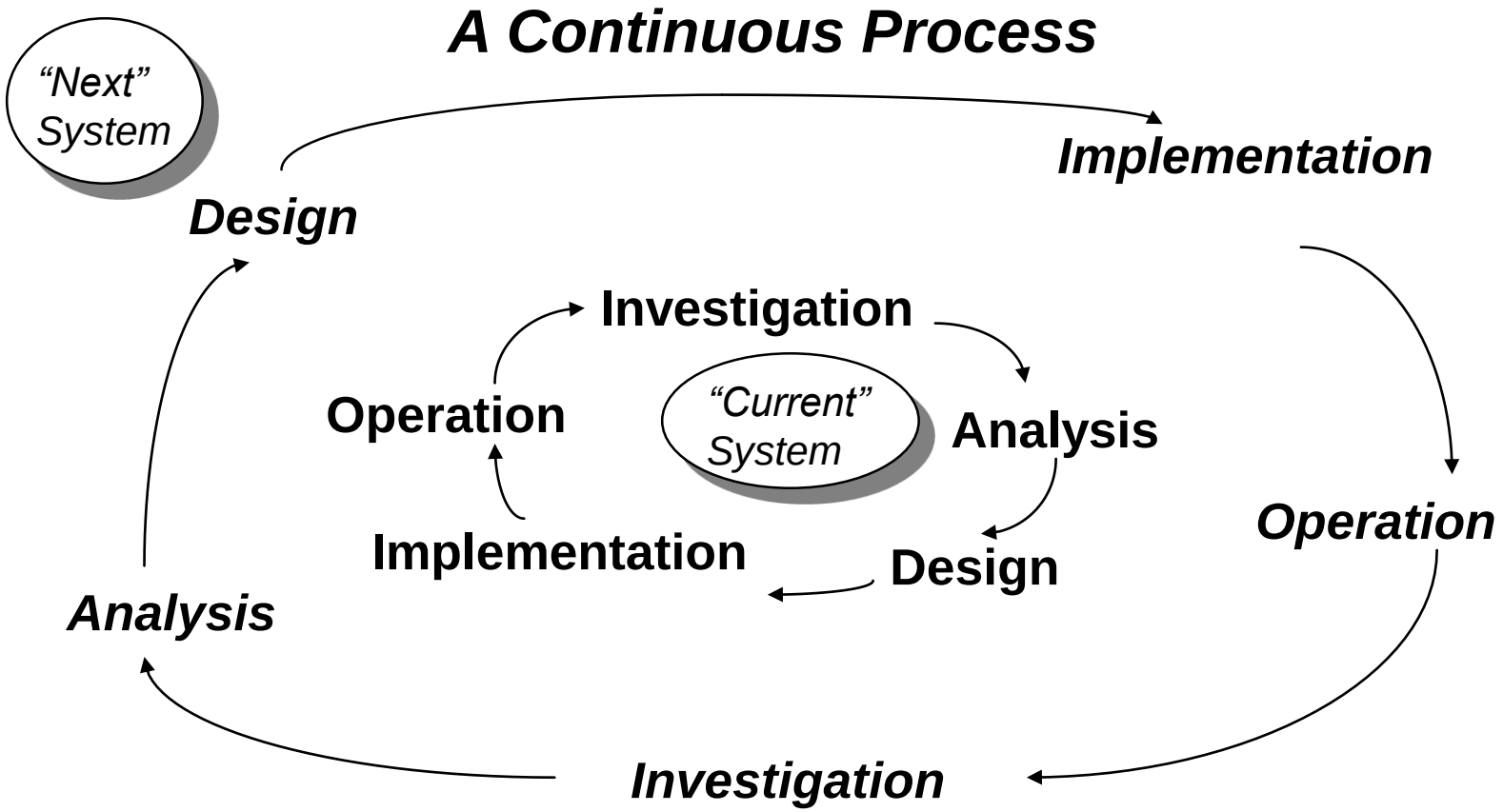
Frequently asked questions about software engineering

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, software development, software validation and software 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.

System Lifecycle



Software process activities

- **A software process is a set of activities and their output, which result in a software product:**

Requirements and specification

Includes

- Domain analysis
- Defining the problem
- Requirements gathering
  - » Obtaining input from as many sources as possible
- Requirements analysis
  - » Organizing the information
- Requirements specification
  - » Writing detailed instructions about how the software should behave

Software process activities

Design

Deciding how the requirements should be implemented, using the available technology

Includes:

- Systems engineering: Deciding what should be in hardware and what in software
- Software architecture: Dividing the system into subsystems and deciding how the subsystems will interact
- Detailed design of the internals of a subsystem
- User interface design
- Design of databases

Software process activities

Implementation

Translate designs into a working system

- Coding
- Testing
- Documentation
- Data conversion (from old to new system)
- Training
- Installation

Software process activities

Maintenance: Evolving system

- Requirements WILL CHANGE to reflect dynamic environment of business

- Continuous process

- Maintenance types:

  Corrective: correct existing defects

  Perfective: improve

  Adaptive: to new environment / requirements

Software Process model

- An abstract representation of a software process, presented from a particular perspective; for example, workflow (sequence of activities), data-flow (information flow), or role/action (who does what)

- These process models explain different approaches to software development; for example, Waterfall, Iterative, and Component Based Software Engineering

Software Engineering methods

- Structured approaches to software development, including:
  - Model descriptions: Describes graphical models (i.e. object, data-flow, state machine models, etc)
  - Rules: Constraints applied to system models (i.e. entities must have unique names)
  - Recommendations: Best practices for designing software (i.e. include no more than nine processes in a data flow diagram)
  - Process guidance: what activities to follow (i.e. document object attributes before defining its operations)
- Examples of methods:
  - Functional oriented: DeMarco's Structured Analysis and Jackson's JSD
  - Object oriented: Booch, Rumbaugh, and Boehm's Obejct Oriented methods, Rational Unified Process

CASE: Computer-Aided Software Engineering

- **Computer-aided software engineering (CASE) is software to support software development and evolution processes.**
- **Activity automation**
  - Graphical editors for system model development;
  - Data dictionary to manage design entities;
  - Graphical UI builder for user interface construction;
  - Debuggers to support program fault finding;
  - Automated translators to generate new versions of a program.

Key points [Professional Software Development]

- Software engineering is an engineering discipline that is concerned with all aspects of software production
- Essential software product attributes are maintainability, dependability and security, efficiency and acceptability
- The high-level activities of specification, development, validation and evolution are part of all software processes
- The fundamental notions of software engineering are universally applicable to all types of system development

Key points [Professional Software Development]

- There are many different types of system and each requires appropriate software engineering tools and techniques for their development

- The fundamental ideas of software engineering are applicable to all types of software system

Software engineering ethics

- Software engineering involves wider responsibilities than simply the application of technical skills
- Software engineers must behave in an honest and ethically responsible way if they are to be respected as professionals
- Ethical behaviour is more than simply upholding the law but involves following a set of principles that are morally correct

Issues of professional responsibility

✧ Confidentiality

- Engineers should normally respect the confidentiality of their employers or clients irrespective of whether or not a formal confidentiality agreement has been signed

✧ Competence

- Engineers should not misrepresent their level of competence. They should not knowingly accept work which is outwith their competence

Issues of professional responsibility

✧ Intellectual property rights

- Engineers should be aware of local laws governing the use of intellectual property such as patents, copyright, etc. They should be careful to ensure that the intellectual property of employers and clients is protected.

✧ Computer misuse

- Software engineers should not use their technical skills to misuse other people's computers. Computer misuse ranges from relatively trivial (game playing on an employer's machine, say) to extremely serious (dissemination of viruses).

ACM/IEEE Code of Ethics

- The professional societies in the US have cooperated to produce a code of ethical practice
- Members of these organisations sign up to the code of practice when they join
- The Code contains eight Principles related to the behaviour of and decisions made by professional software engineers, including practitioners, educators, managers, supervisors and policy makers, as well as trainees and students of the profession

The ACM/IEEE Code of Ethics

Software Engineering Code of Ethics and Professional Practice

ACM/IEEE-CS Joint Task Force on Software Engineering Ethics and Professional Practices

PREAMBLE

The short version of the code summarizes aspirations at a high level of the abstraction; the clauses that are included in the full version give examples and details of how these aspirations change the way we act as software engineering professionals. Without the aspirations, the details can become legalistic and tedious; without the details, the aspirations can become high sounding but empty; together, the aspirations and the details form a cohesive code.

Software engineers shall commit themselves to making the analysis, specification, design, development, testing and maintenance of software a beneficial and respected profession. In accordance with their commitment to the health, safety and welfare of the public, software engineers shall adhere to the following Eight Principles:

Ethical principles

1. PUBLIC - Software engineers shall act consistently with the public interest.
2. CLIENT AND EMPLOYER - Software engineers shall act in a manner that is in the best interests of their client and employer consistent with the public interest.
3. PRODUCT - Software engineers shall ensure that their products and related modifications meet the highest professional standards possible.
4. JUDGMENT - Software engineers shall maintain integrity and independence in their professional judgment.
5. MANAGEMENT - Software engineering managers and leaders shall subscribe to and promote an ethical approach to the management of software development and maintenance.
6. PROFESSION - Software engineers shall advance the integrity and reputation of the profession consistent with the public interest.
7. COLLEAGUES - Software engineers shall be fair to and supportive of their colleagues.
8. SELF - Software engineers shall participate in lifelong learning regarding the practice of their profession and shall promote an ethical approach to the practice of the profession.