LECTURE (1)
INTRODUCTION TO SOFTWARE ENGINEERING

AMR E. MOHAMED
Faculty of Engineering - Helwan University

لما نصدق اننا نِقْدَر

نِـقْدَر ،،،

OUTLINE

- Introduction to Software
- Introduction to Software Engineering
- Software Components
- Software Process
- Classification of Programming Techniques

INTRODUCTION TO SOFTWARE

INTRODUCTION TO SOFTWARE

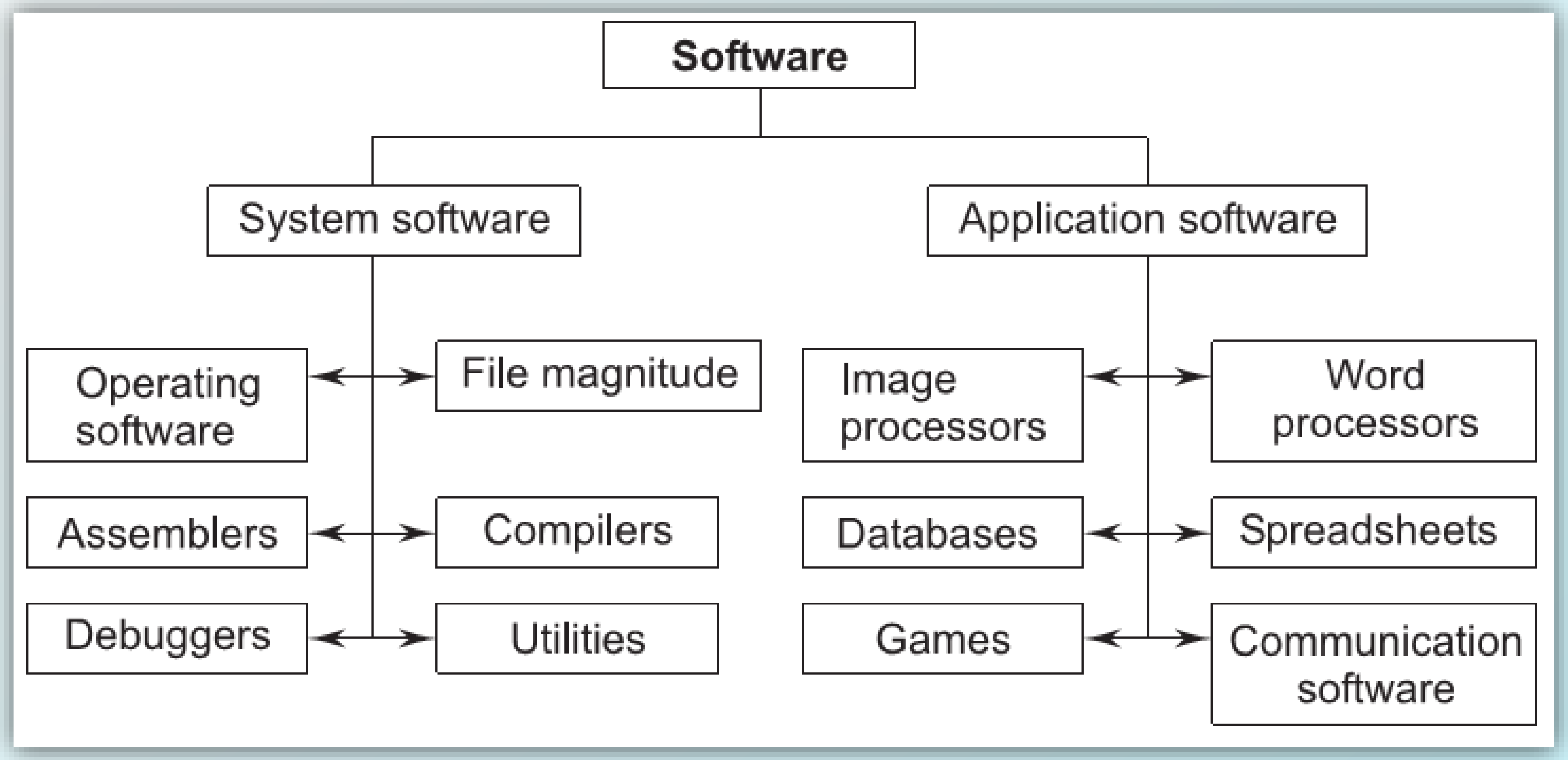
- Definition of Software:
  - Software includes
    - Computer programs (**Instructions**) that when executed provide desired functions and performance.
    - Data structures that enable the programs to adequately manipulate information.
    - Documents that describe the operation and use of the programs.
- Software is ~10x more expensive to produce than a computer program [Brooks75]

IMPORTANCE OF SOFTWARE

- Computer software has become a driving force.
  - It is the engine that drives **business decision making**.
  - It serves as the basis for modern **scientific investigation** and **engineering problem-solving**.
  - It is **embedded** in all kinds of systems, such as transportation, medical, telecommunications, military, industrial processes, entertainment, office products, etc.

TYPES OF SOFTWARE

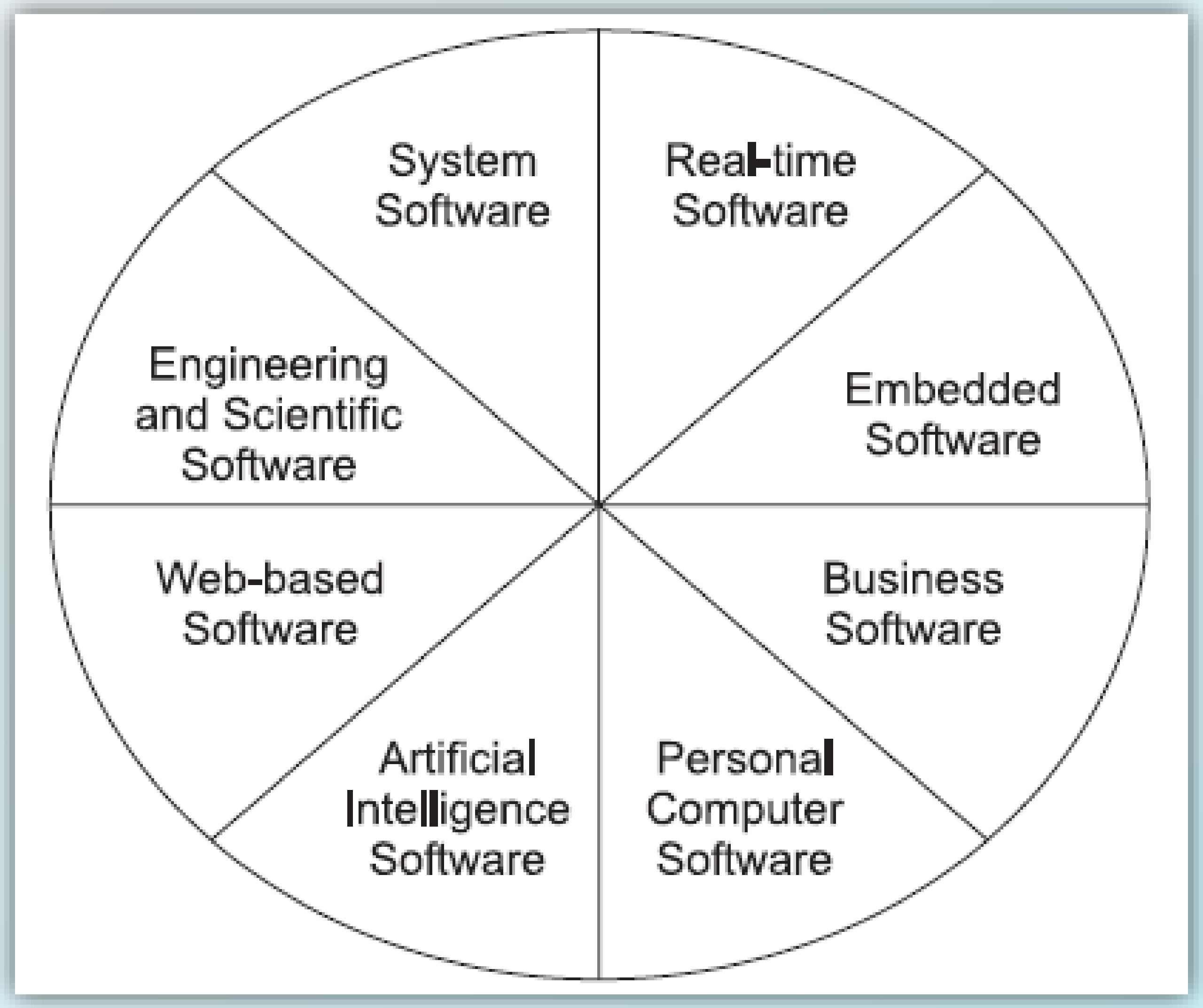
- Computer software is often divided into two categories:



CLASSES OF SOFTWARE

- Software is classified into the following two classes:
  - Generic Software.
    - 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.
  - Customized Software.
    - Software that is commissioned by a specific customer to meet their own needs.
    - **Examples** – embedded control systems, air traffic control software, traffic monitoring systems.

SOFTWARE APPLICATIONS


System Software
Real-time Software
Engineering and Scientific Software
Embedded Software
Web-based Software
Business Software
Artificial Intelligence Software
Personal Computer Software

INTRODUCTION TO SOFTWARE ENGINEERING

DEFINITION OF SOFTWARE ENGINEERING

- **Software Engineering** is a discipline whose aim is the production of fault free software that satisfies the user’s needs and that is delivered on time and within budget

- **Software Engineering** is a collection of techniques, methodologies and tools that help with the production of A high quality software system developed with a given budget before a given deadline while change occurs

SOFTWARE CRISIS

- The software crisis has been with us since the 1970s. As per the latest IBM report,
  - "31% of the projects get cancelled before they are completed,
  - 53% over-run their cost-estimates by an average of 189% ,
  - and for every 100 projects, there are 94 restarts."

SOFTWARE

- Size: large
- User is not the developer
- Lifespan: long (no ageing)
- Cost: development + operation/maintenance

SOFTWARE COMPLEXITY

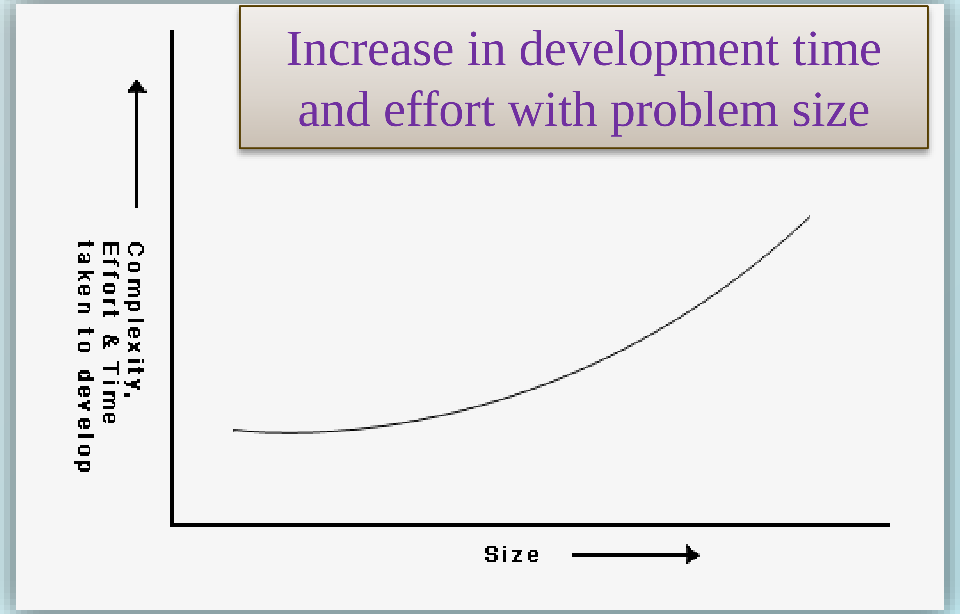
- Kitchen table: 5-10 components
- Bicycle: 20-100 components
- Car: 30.000 parts/components
- Airplane: 100.000 parts/components.
- Cell phone Software, printer driver Software : 1M Lines of code
- Cellular network Software, operating system: several Millions

SOFTWARE COMPLEXITY

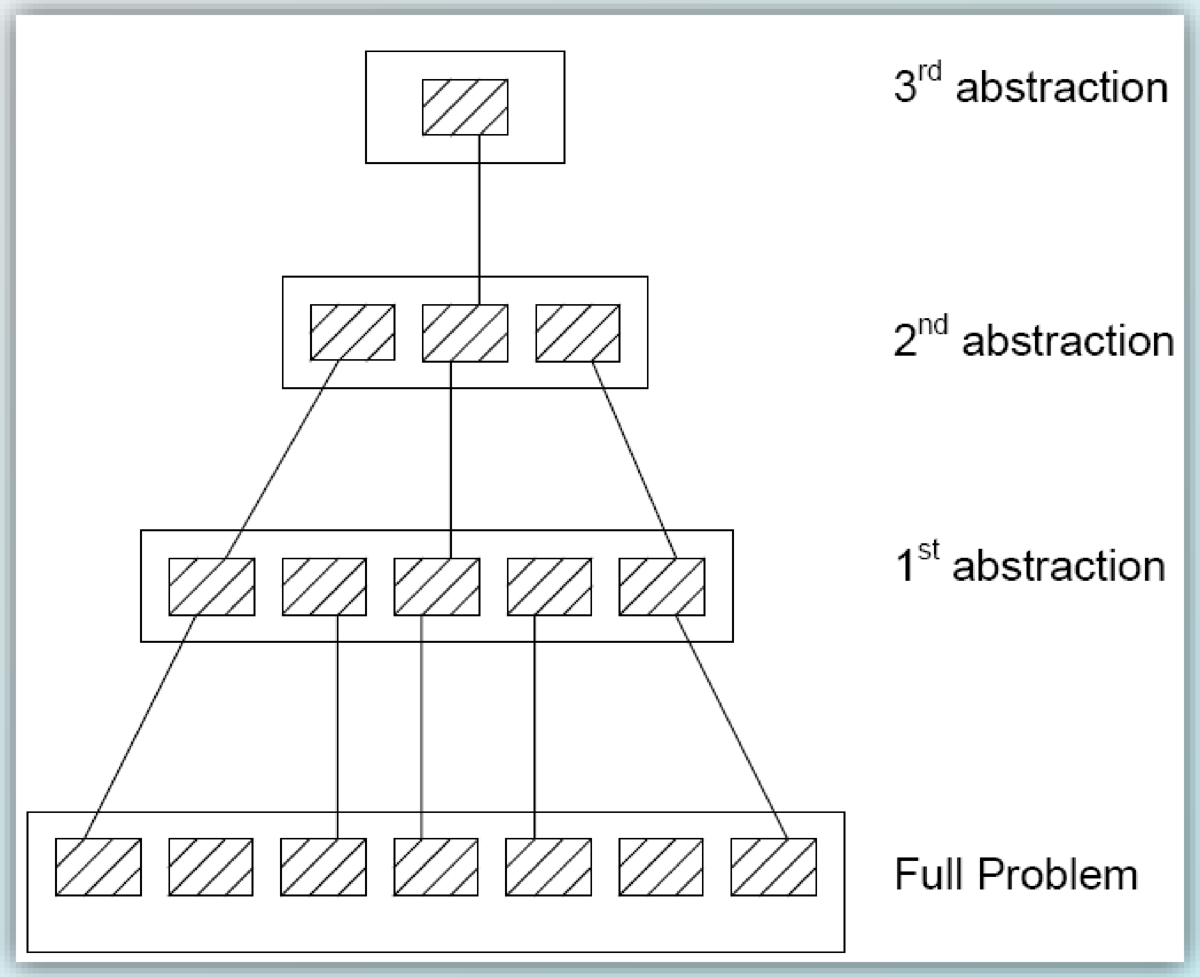
- Windows 7 about 50 millions lines of code.
  - http://answers.yahoo.com/question/index?qid=20080712132328AAwyert
- An Android operating system in a smart phone consists of 12 million lines of code.
  - https://docs.google.com/viewer?url=http%3A%2F%2Fwww.rttonline.co
- Boeing's new 787 Dreamliner requires about 6.5 million lines of software code to operate its avionics and onboard support systems.
  - http://spectrum.ieee.org/green-tech/advanced-cars/this-car-runs-on-co

SCOPE AND NECESSITY OF SOFTWARE ENGINEERING

- ❑ Software engineering helps to reduce the programming complexity.
- ❑ software products have to easy to
  - ☑ Alter ☑ Debug ☑ Enhance
  - ☑ Use resources optimally ☑ Meet the user requirements.
- ❑ Software engineering principles use two important techniques to deal with Complexity (reduce problem complexity):
  - Abstraction.
  - Decomposition

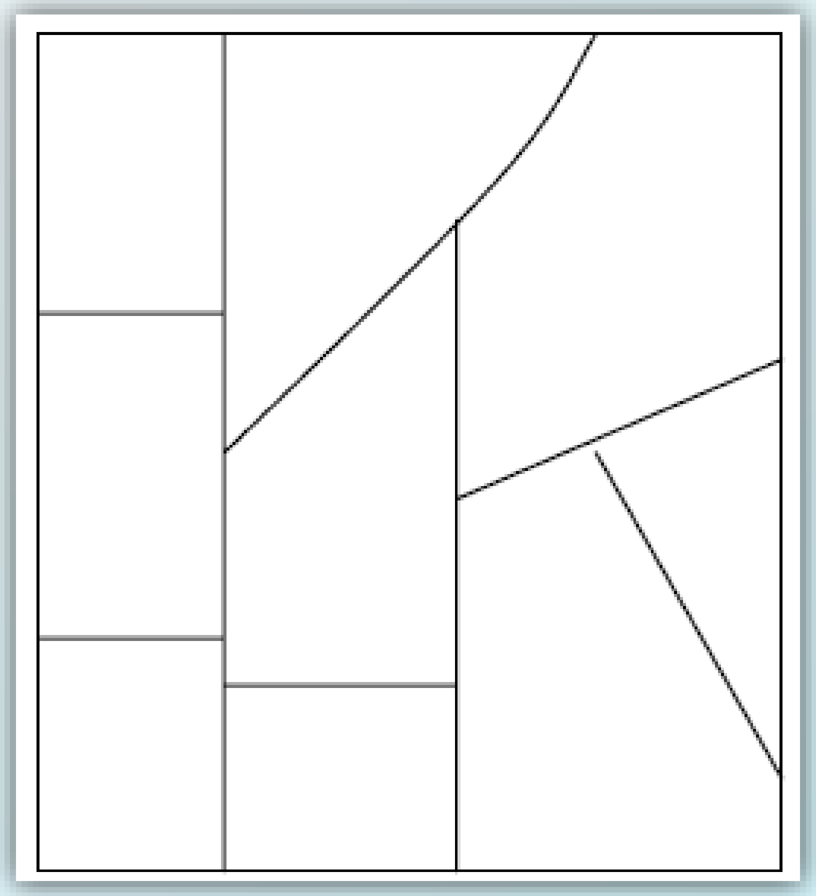


ABSTRACTION



A hierarchy of abstraction

DECOMPOSITION



Decomposition of a large problem into a set of smaller problems

PROGRAM VS. SOFTWARE PRODUCT

- Programs
  - are developed by individuals for their personal use
  - are Small in size
  - have limited functionality
  - the programmer himself is the sole user
  - a single developer is involved
  - the user interface may not be very important
  - very little documentation is expected
  - can be developed according to the programmer’s individual style of development

SOFTWARE

- Software is described by
  - Its capabilities
    - Functions it executes.
    - Features it provides.
    - Facilities it offers.
  - The platform specifications that are required to run it.
    - Certain hardware
    - Certain operating system

ATTRIBUTES OF GOOD SOFTWARE

- **Maintainability**: software should be written in such a way that it may evolve to meet the changing needs of customer.
- **Dependability**: software dependability has a range of characteristics, including reliability, security and safety.
- **Efficiency**: efficiency includes responsiveness, processing time, memory utilization etc…
- **Usability**: it should have an appropriate user interface and adequate documentation.

MISCONCEPTIONS

- Software is free
  - a medium sized project with 50.000 LOC costs between $400.000 to $1.600.000 in personnel
- Software is soft
  - Changing it is difficult and costly than hardware.
  - Cost of maintenance > cost of development.
  - Maintenance becomes impossible at a certain point
- Software is produced
  - Software is not mass produced (like machines)
  - Software is developed
- Software ages
  - Failures do not occur due to material fatigue (as with hardware) but due to the execution of logical faults
  - Software changes due to requirements changes, platform changes.

ROLE OF SOFTWARE ENGINEER

- Programming skill not enough
- Software engineering involves "programming-in the-large"
  - Understand requirements and write specifications
    - Derive models and reason about them
  - Master software
  - Operate at various abstraction levels
  - Member of a team
    - Communication skills
    - Management skills

THE NEED FOR DISCIPLINED PRACTICES

- The job of software engineers is to
  - produce quality products
  - produce them on schedule
  - and do this work for the planned costs

- In this class I hope we will learn some of this
  - You will also need a lot of practice!!

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 COMPONENTS

SOFTWARE COMPONENTS

- A software component is a system element offering a predefined service and is able to communicate with other components.

- The Software component shall be
  - Multiple-use
  - Non-context-specific
  - Compassable with other components
  - Encapsulated, i.e., non-investigable through its interfaces
  - A unit of independent deployment and versioning

SOFTWARE COMPONENTS

- The component needs
  - to be fully documented.
  - to be more thoroughly tested.
  - to have robust input validity checking.
  - to pass back useful error messages as appropriate.
  - to be built with an awareness that it will be put to unforeseen uses.
  - a mechanism for compensating developers who invest the (substantial) effort implied above.

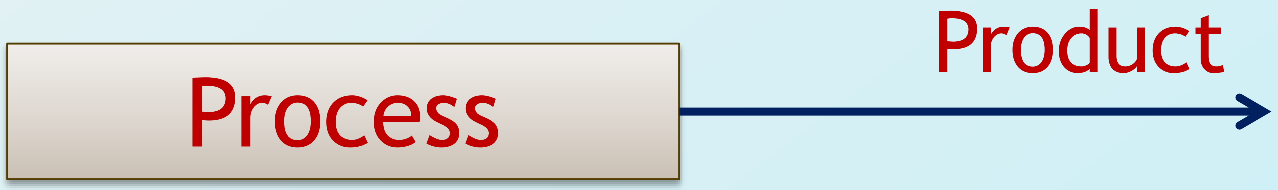
SOFTWARE PROCESS

SOFTWARE PROCESS

- For success in large software development, It is important to follow an Engineering approach, consisting of a well-defined process.
- A process: is a series of steps involving activities, constraints, and resources that produce an intended output of some kind.
- A process involves a set of tools and techniques.
- A software process: is the related set of activities and processes that are involved in developing and evolving a software system.
- A Software process: deals with both technical and management issues.

SOFTWARE PROCESS

- Process: activities, people, tools
- Products: documents, data, code.
- The quality of the product depends on the quality of the process



SOFTWARE PROCESSES

- Software specifications: The functionality of the software and constraints on its operation must be defined.
- Software development: Software that meets the specifications must be produced.
- Software validation: The software must be validated to ensure that it does what the customer wants.
- Software evolution: The software must evolve to meet changing customer needs.

CLASSIFICATION OF PROGRAMMING TECHNIQUES

CLASSIFICATION OF PROGRAMMING TECHNIQUES

- Unstructured programming. [use of GOTO]
- Procedural programming.
- Modular programming.
- Object-oriented programming.

IMPORTANT ADVANTAGES OF STRUCTURED PROGRAMMING

- Structured programs are easier to
  - easier to read and understand.
  - easier to maintain.
  - require less effort and time for development.
  - easier debugging and usually fewer errors are made in the course of writing such programs.

EVOLUTION OF SOFTWARE DESIGN TECHNIQUES OVER THE LAST 50 YEARS

- Assembly language.
- High-level language programming. (FORTRAN, ALGOL, and COBOL)
- Control flow structure design, and the use of "GOTO" statement ➔ Jump.
- Control flow structure design, and the use of (selection, sequence and iteration)
- Data structure-oriented design.
- Data flow-oriented design technique.
- Object-oriented design.

EXPLORATORY STYLE VS. MODERN STYLE OF SOFTWARE DEVELOPMENT

- Exploratory software development style is based on error correction
  - developing a working system as quickly as possible and then successively modifying it until it performed satisfactorily.
- The software engineering principles are primarily based on error prevention

CASE STUDIES

CASE STUDIES

- A personal insulin pump
  - An embedded system in an insulin pump used by diabetics to maintain blood glucose control.
- A mental health case patient management system
  - A system used to maintain records of people receiving care for mental health problems.

INSULIN PUMP CONTROL SYSTEM

- Collects data from a blood sugar sensor and calculates the amount of insulin required to be injected.
- Calculation based on the rate of change of blood sugar levels.
- Sends signals to a micro-pump to deliver the correct dose of insulin.
- Safety-critical system as low blood sugars can lead to brain malfunctioning, coma and death; high-blood sugar levels have long-term consequences such as eye and kidney damage.

INSULIN PUMP HARDWARE ARCHITECTURE

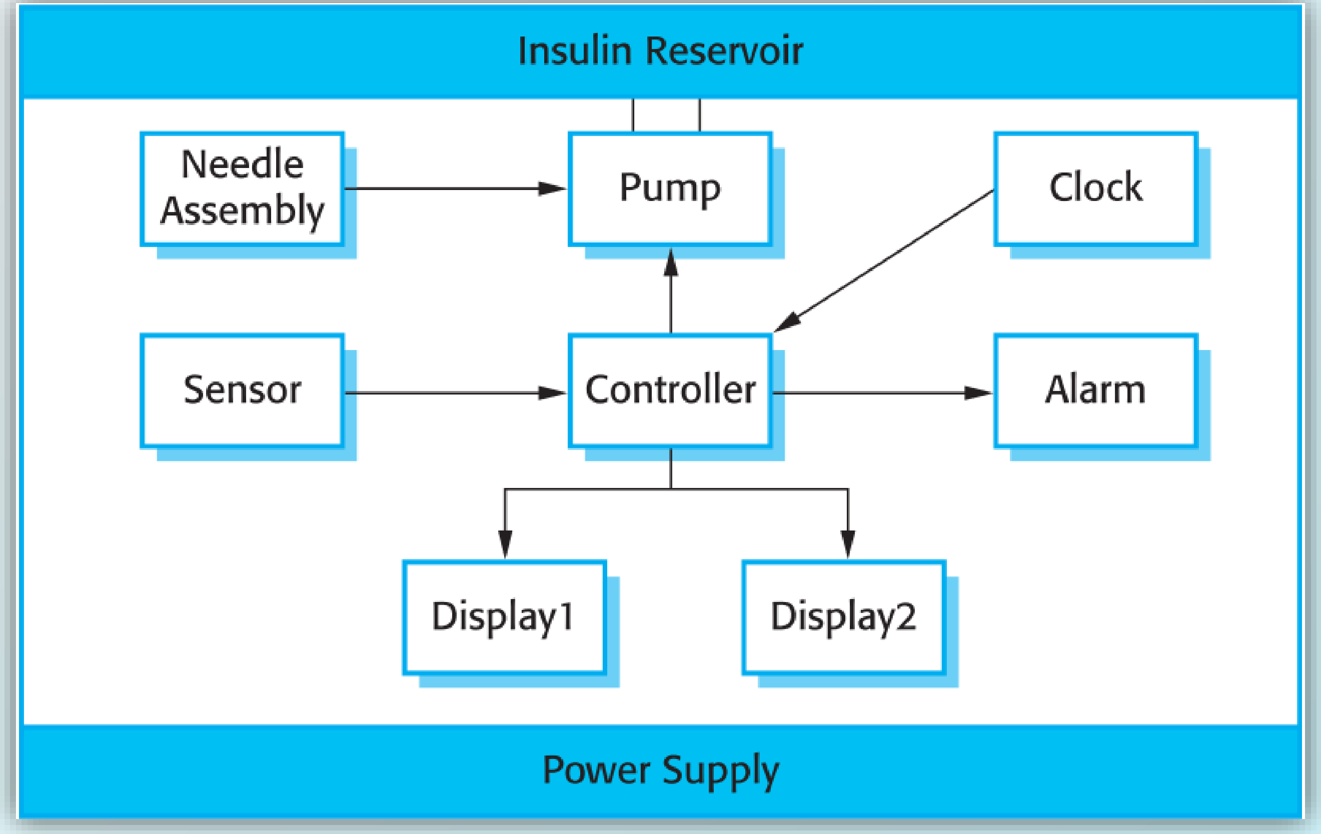


Figure 1.4: Insulin pump hardware

ACTIVITY MODEL OF THE INSULIN PUMP

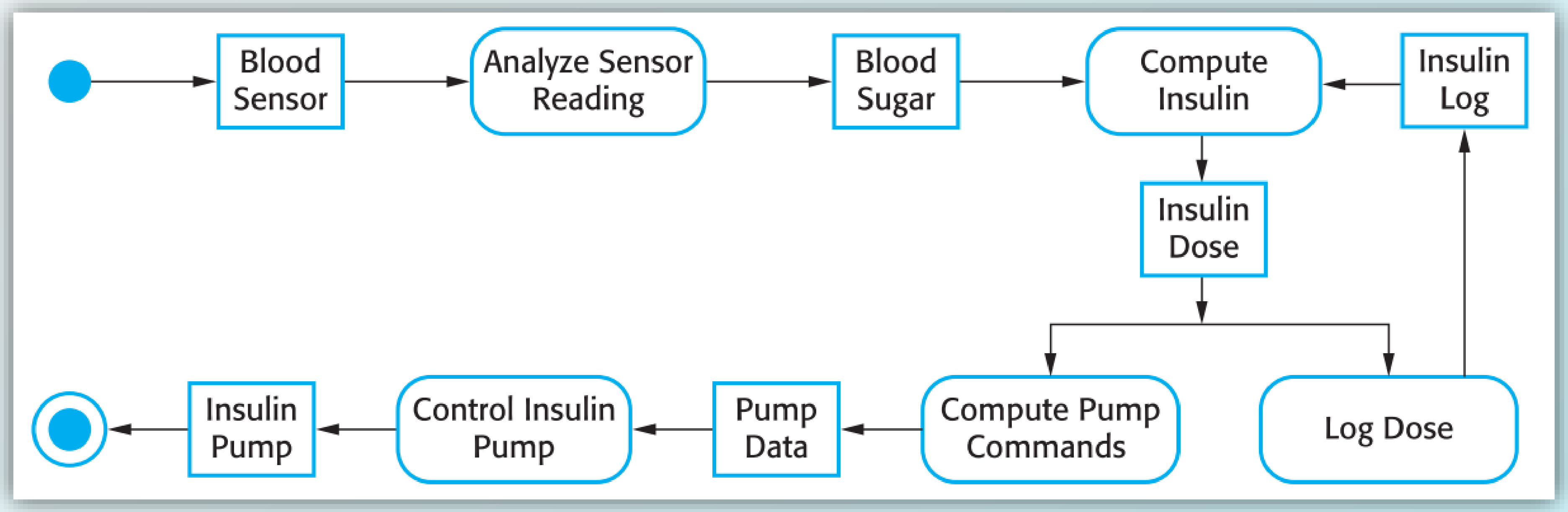


Figure 1.5: Activity model of the insulin pump

ESSENTIAL HIGH-LEVEL REQUIREMENTS

- The system shall be available to deliver insulin when required.
- The system shall perform reliably and deliver the correct amount of insulin to counteract the current level of blood sugar.
- The system must therefore be designed and implemented to ensure that the system always meets these requirements.

A PATIENT INFORMATION SYSTEM FOR MENTAL HEALTH CARE

- A patient information system to support mental health care is a medical information system that maintains information about patients suffering from mental health problems and the treatments that they have received.
- Most mental health patients do not require dedicated hospital treatment but need to attend specialist clinics regularly where they can meet a doctor who has detailed knowledge of their problems.
- To make it easier for patients to attend, these clinics are not just run in hospitals. They may also be held in local medical practices or community centres.

MHC-PMS

- The MHC-PMS (Mental Health Care-Patient Management System) is an information system that is intended for use in clinics.
- It makes use of a centralized database of patient information but has also been designed to run on a PC, so that it may be accessed and used from sites that do not have secure network connectivity.
- When the local systems have secure network access, they use patient information in the database but they can download and use local copies of patient records when they are disconnected.

MHC-PMS GOALS

- To generate management information that allows health service managers to assess performance against local and government targets.
- To provide medical staff with timely information to support the treatment of patients.

THE ORGANIZATION OF THE MHC-PMS

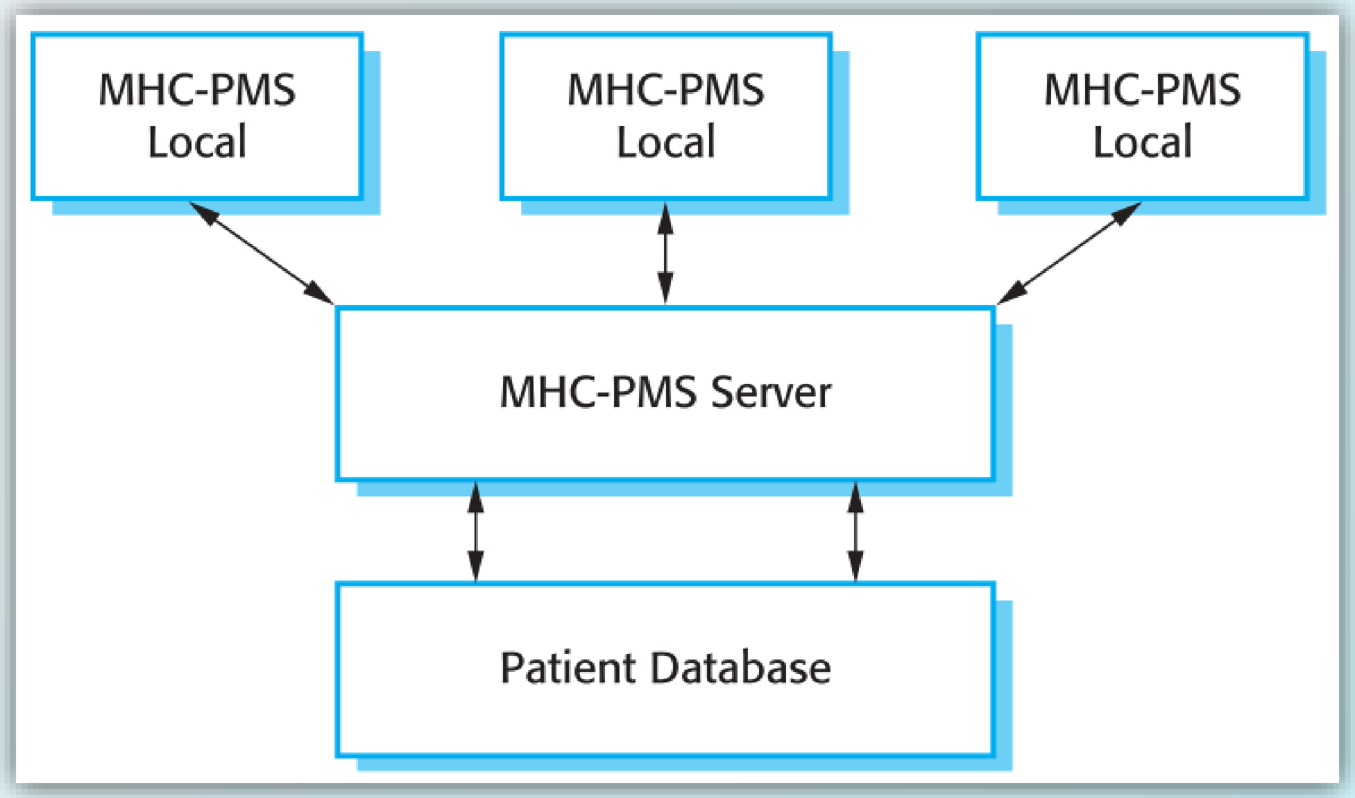


Figure 1.6: The organization of the MHC-PMS

MHC-PMS KEY FEATURES

- Individual care management
  - Clinicians can create records for patients, edit the information in the system, view patient history, etc. The system supports data summaries so that doctors can quickly learn about the key problems and treatments that have been prescribed.
- Patient monitoring
  - The system monitors the records of patients that are involved in treatment and issues warnings if possible problems are detected.
- Administrative reporting
  - The system generates monthly management reports showing the number of patients treated at each clinic, the number of patients who have entered and left the care system, number of patients sectioned, the drugs prescribed and their costs, etc.

MHC-PMS CONCERNS

- Privacy
  - It is essential that patient information is confidential and is never disclosed to anyone apart from authorised medical staff and the patient themselves.
- Safety
  - Some mental illnesses cause patients to become suicidal or a danger to other people. Wherever possible, the system should warn medical staff about potentially suicidal or dangerous patients.
  - The system must be available when needed otherwise safety may be compromised and it may be impossible to prescribe the correct medication to patients.

Thank you
for
your attention