Software Engineering Lecture 2

By

Engr, Sara Rehmat

MS (CS)

Types of Software

- Stand-alone applications
- Interactive transaction-based applications
- Embedded control systems
- Batch processing systems
- Entertainment systems
- Systems for modeling and simulation
- Data Collection Systems
- Systems of systems

Types of Software

- Legacy Software
- WebApps

Legacy Software

- An old and outdated program that is still used, even though newer and efficient options are available
- Characterized by longevity and business criticality.
- If the legacy software meets the needs of its users and runs reliably, no need to be fixed.
- May have low quality and interoperability issues with other software

Example of legacy software

The Space Shuttle was low Earth orbital Spacecraft operated from 1981 to 2011 by NASA.
https://en.wikipedia.org/wiki/Legacy_system#NASA_example

WebApps

- The Web is now a platform for running application and organizations are increasingly developing web-based systems rather than local systems.
- Cloud computing is an approach to the provision of computer services where applications run remotely on the ‘cloud’.
- The practice of using a network of remote servers hosted on the Internet to store, manage, and process data, rather than a local server or a personal computer.

Characteristics of WebApps - I

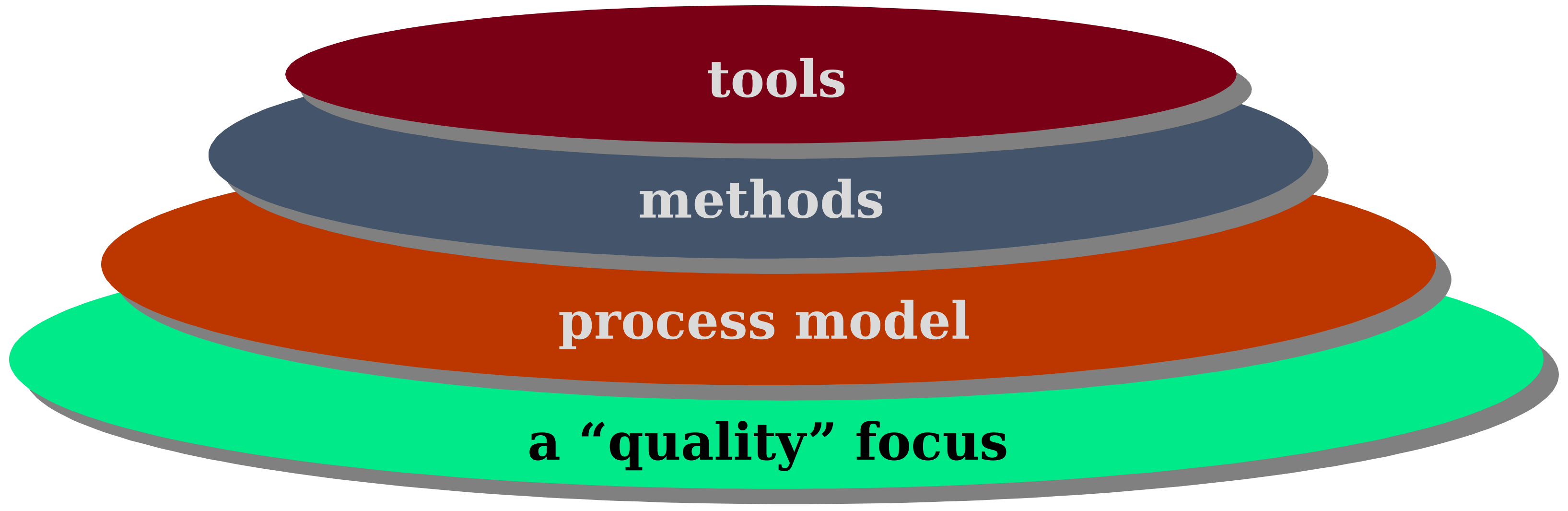
- **Network intensiveness.** A WebApp resides on a network and must serve the needs of a diverse community of clients.
- **Concurrency.** A large number of users may access the WebApp at one time.
- **Unpredictable load.** The number of users of the WebApp may vary by orders of magnitude from day to day.
- **Performance.** If a WebApp user must wait too long (for access, for server-side processing, for client-side formatting and display), he or she may decide to go elsewhere.
- **Availability.** Although expectation of 100 percent availability is unreasonable, users of popular WebApps often demand access on a “24/7/365” basis.

Characteristics of WebApps - III

- **Data driven.** The primary function of many WebApps is to use hypermedia to present text, graphics, audio, and video content to the end-user.
- **Content sensitive.** The quality and aesthetic nature of content remains an important determinant of the quality of a WebApp.
- **Continuous evolution.** Unlike conventional application software that evolves over a series of planned, chronologically-spaced releases, Web applications evolve continuously.

- **Immediacy.** Although *immediacy*—the compelling need to get software to market quickly—is a characteristic of many application domains, WebApps often exhibit a time to market that can be a matter of a few days or weeks.
- **Security.** Because WebApps are available via network access, it is difficult, if not impossible, to limit the population of end-users who may access the application.
- **Aesthetics.** An undeniable part of the appeal of a WebApp is its look and feel.

A Layered Technology

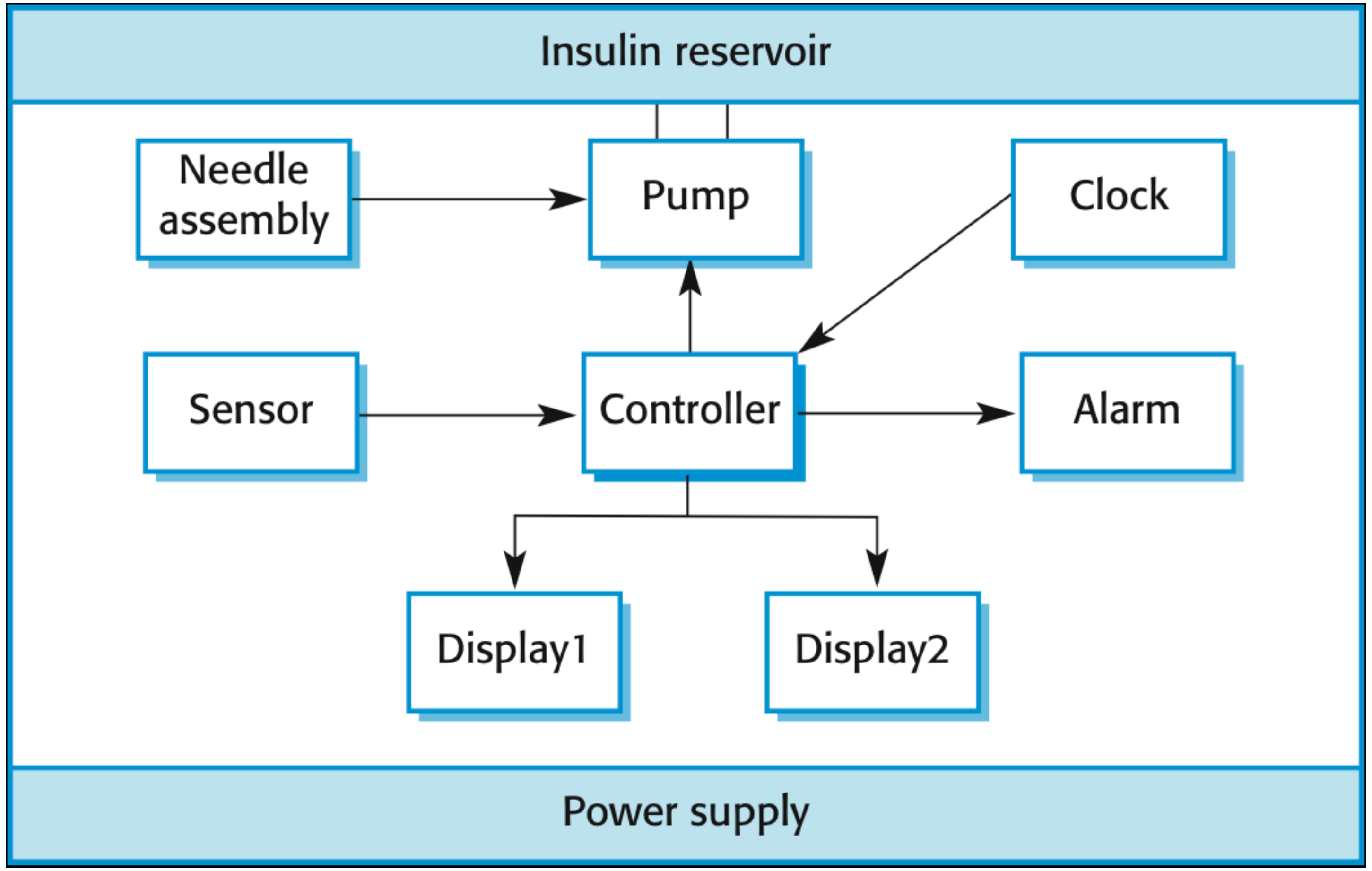


Software Engineering

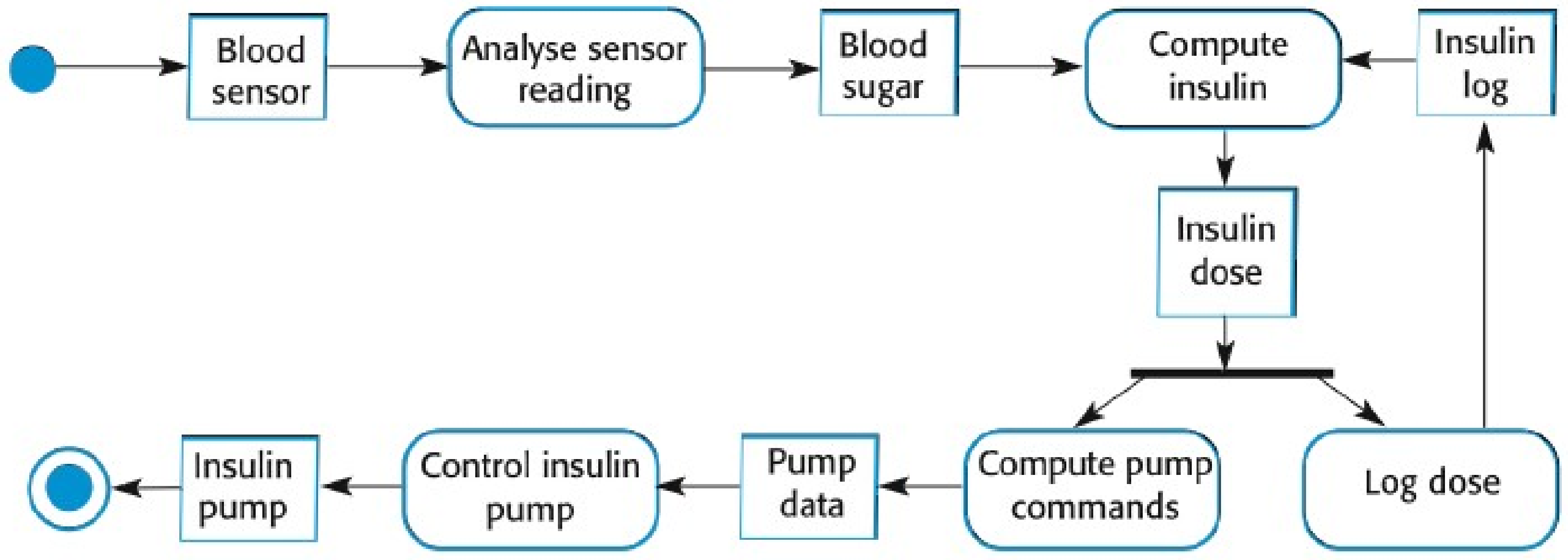
Case Studies

- A case study is a particular instance of something used or analyzed in order to illustrate a thesis or principle.
- An embedded insulin pump control system
- A system for mental health care patient management
- A wilderness weather station

A personal insulin pump



A personal insulin pump – Activity Model



Essential high-level requirements of Insulin - Pump

- 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.

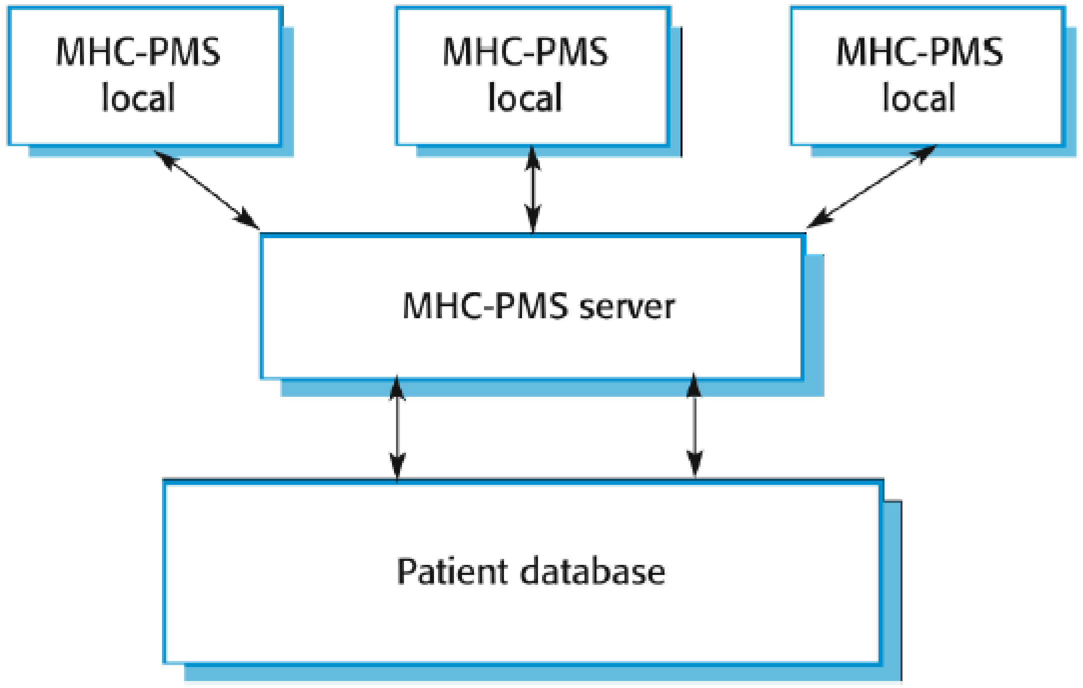
MHC-PMS

- 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



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.

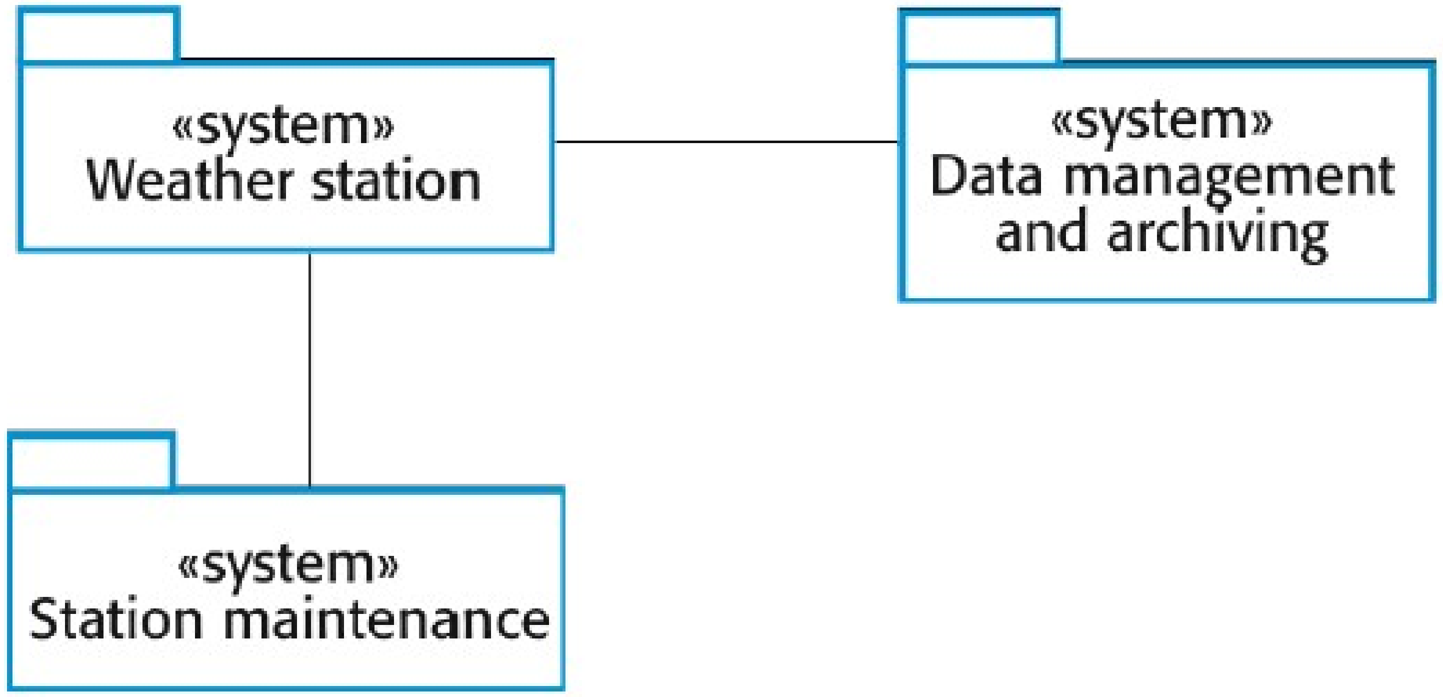
MHC-PMS concerns

- **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 thecorrect medication to patients

Wilderness Weather Station

- The government of a country with large areas of wilderness decides to deploy several hundred weather stations in remote areas .
- Weather stations collect data from a set of instruments that measure temperature and pressure, sunshine, rainfall, wind speed and wind direction
- The weather station includes a number of instruments that measure weather parameters such as the wind speed and direction, the ground and air temperatures, the barometric pressure and the rainfall over a 24-hour period.
- Each of these instruments is controlled by a software system that takes parameter readings periodically and manages the data collected from the instruments.

The weather station’s environment



Weather information system

- **The weather station system**
- This is responsible for collecting weather data, carrying out some initial data processing and transmitting it to the data management system.
- **The data management and archiving system**
- This system collects the data from all of the wilderness weather stations, carries out data processing and analysis and archives the data.
- **The station maintenance system**
- This system can communicate by satellite with all wilderness weather stations to monitor the health of these systems and provide reports of problems.

Additional software functionality

- Monitor the instruments, power and communication hardware and report faults to the management system.
- Manage the system power, ensuring that batteries are charged whenever the environmental conditions permit but also that generators are shut down in potentially damaging weather conditions, such as high wind.
- Support dynamic reconfiguration where parts of the software are replaced with new versions and where backup instruments are switched into the system in the event of system failure.

Software Engineering Principles

- Polya suggests:
    1. Understand the problem (communication and analysis).
    2. Plan a solution (modeling and software design).
    3. Carry out the plan (code generation).
    4. Examine the result for accuracy (testing and quality assurance).

1. Understand the Problem

- *Who has a stake in the solution to the problem?* That is, who are the stakeholders?
- *What are the unknowns?* What data, functions, and features are required to properly solve the problem?
- *Can the problem be compartmentalized?* Is it possible to represent smaller problems that may be easier to understand?
- *Can the problem be represented graphically?* Can an analysis model be created?

Plan the Solution

- *Have you seen similar problems before?* Are there patterns that are recognizable in a potential solution? Is there existing software that implements the data, functions, and features that are required?
- *Has a similar problem been solved?* If so, are elements of the solution reusable?
- *Can subproblems be defined?* If so, are solutions readily apparent for the subproblems?
- *Can you represent a solution in a manner that leads to effective implementation?* Can a design model be created?

Carry Out the Plan

- *Does the solution conform to the plan?* Is source code traceable to the design model?
- *Is each component part of the solution provably correct?* Has the design and code been reviewed, or better, have correctness proofs been applied to algorithm?

Examine the Result

- *Is it possible to test each component part of the solution?* Has a reasonable testing strategy been implemented?
- *Does the solution produce results that conform to the data, functions, and features that are required?* Has the software been validated against all stakeholder requirements?

Software Engineering Ethics

- **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

Software Engineering Ethics contd..

- **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).

References

- Software Engineering, 9th Ed. Ian Sommerville. Pearson Education
- Software Engineering: A Practitioner’s Approach, 6th Ed. Roger S. Pressman, McGraw-Hill, 2009

Questions