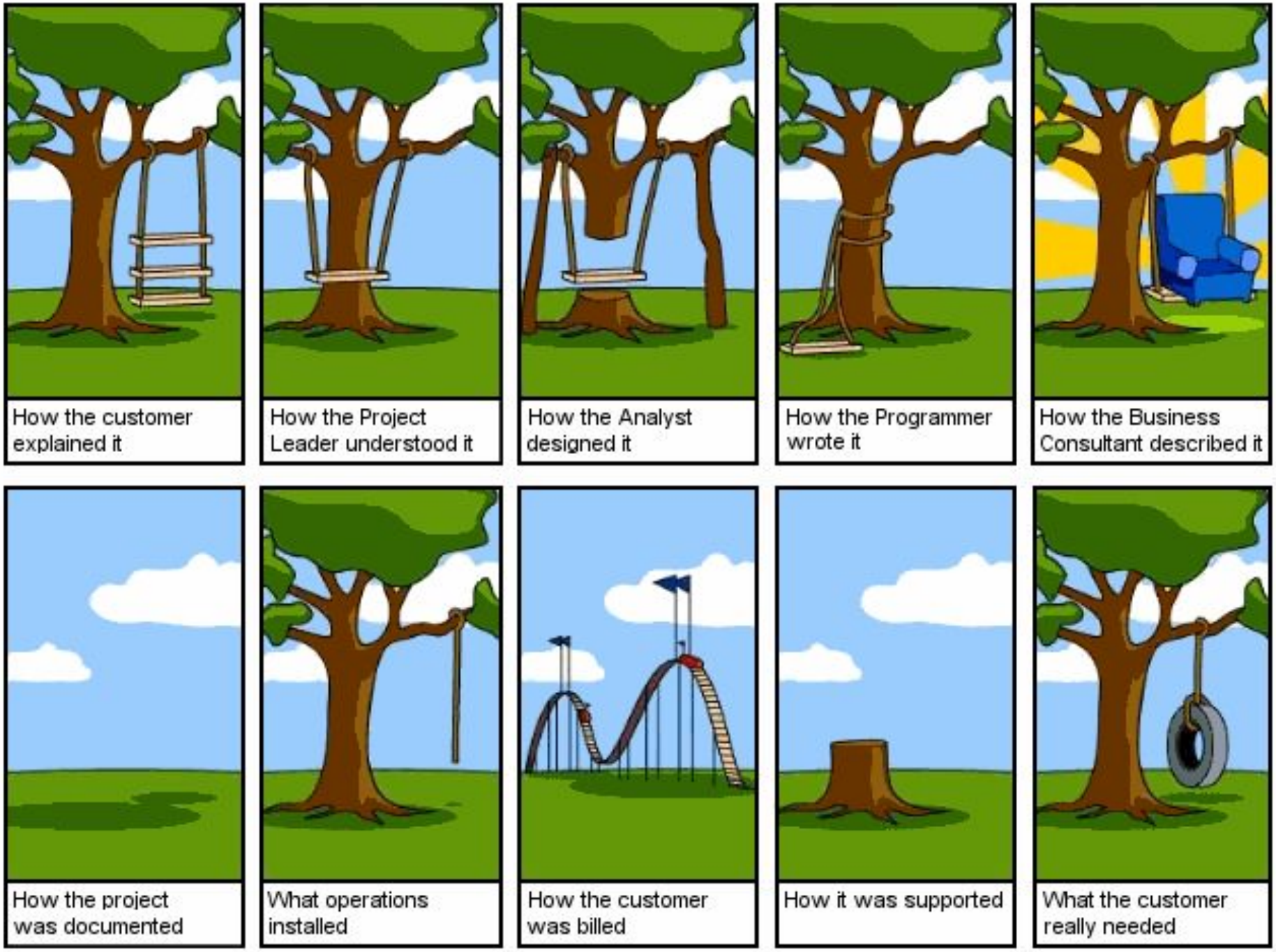
Software Engineering

anwarul@cse.green.edu.bd

Introduction

Slides adapted from **Dr. Kirstie Hawkey**
https://web.cs.dal.ca/~hawkey/3130/


How the customer explained it
How the Project Leader understood it
How the Analyst designed it
How the Programmer wrote it
How the Business Consultant described it
How the project was documented
What operations installed
How the customer was billed
How it was supported
What the customer really needed

This Course

- Software Engineering is the systematic approach to development, deployment, operation, maintenance, and retirement of Software (SW).
- Basic Question Software Engineering
  - *How to develop industrial-strength software?*

What this course will give?

- **Main objective:** Give an idea of how industrial-strength software gets developed
- **At the end:** you should have the ability to plan, execute, and manage small software projects.
- **Lectures:** will discuss how to perform different tasks in a project
- **In the project:** the techniques will be applied

Evaluation and Grading

- NO LATE ASSIGNMENTS ACCEPTED EXCEPT IN EXTRAORDINARY SITUATIONS
- Acknowledge all collaborators and any other sources used in all submitted work.
- Plagiarism and other anti-intellectual behavior will be dealt with severely.
- When in doubt, attribute.

Review of…

- Website
- Course Schedule

Warming up…

Skills Survey

- Name: _____Student #:_____ Email:_____
- Number of years: in Comp Sci & Eng._____, programming: _____.
- Rank you skills in the following areas

where 1=poor, 2=passable, 3=good, 4=strong, and 5=exceptional.

- C Coding skills: 1 2 3 4 5
- Java/C++ Coding skills: 1 2 3 4 5
- DB Development: 1 2 3 4 5
- Web Development: 1 2 3 4 5
- Large Software Design: 1 2 3 4 5
- Software Tools: 1 2 3 4 5
- Project Management: 1 2 3 4 5
- Testing skills: 1 2 3 4 5
- Documentation and Writing: 1 2 3 4 5

Write 1 name from the class you would like to work with:
1)

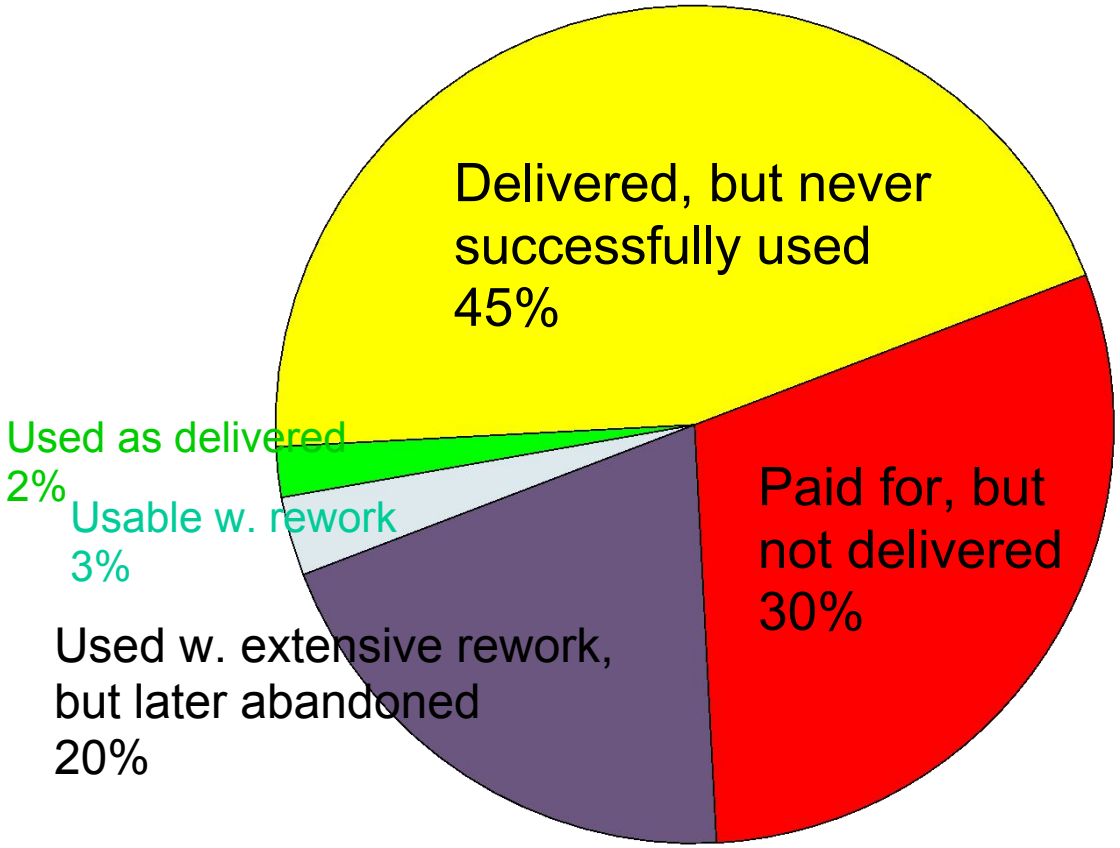
Let’s Begin…..

Contents

- Professional software development
  What is meant by software engineering.
- Software engineering ethics
  A brief introduction to ethical issues that affect software engineering.
- Case studies
  An introduction to three examples that are used in later chapters in the book.

Why Software Engineering?

9 software projects totaling $96.7 million: Where The Money Went
[Report to Congress, Comptroller General, 1979]



Take a look at the Standish Report (The "Chaos" Report)

Software Engineering: A Problem Solving Activity

- **Analysis**: Understand the nature of the problem and break the problem into pieces
- **Synthesis**: Put the pieces together into a large structure

For problem solving we use

- Techniques (methods):
  - Formal procedures for producing results using some well-defined notation
- Methodologies:
  - Collection of techniques applied across software development and unified by a philosophical approach
- Tools:
  - Instrument or automated systems to accomplish a technique

Isn't there something more fundamental than problem "solving"?

FAQs about software engineering

- What is software?
- What is software engineering?
- What is the difference between software engineering and computer science?
- What is a software process model?
- Who does software engineering?
- What are the costs of software engineering?
- What are the attributes of good software?

What is Software?

- Software (IEEE) is a collection of
  - programs,
  - procedures,
  - rules, and
  - associated documentation and data

What is software?

- Computer programs and associated documentation such as requirements, design models and user manuals.
- Software products may be developed for a particular customer or may be developed for a general market.
- Software products may be

Generic - developed to be sold to a range of different customers e.g. PC software such as Excel or Word.

Bespoke (custom) - developed for a single customer according to their specification.

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 organized approach to their work and use appropriate tools and techniques depending on the problem to be solved, the development constraints and the resources available.

Industrial Strength Software

- Student programs != industrial strength software
- Key difference is in quality (including usability, reliability, portability, etc.)
  - High quality requires heavy testing, which consumes roughly 40% (30-50%) of total development effort
  - Requires development be broken in stages such that bugs can be detected in each
  - Good UI, backup, fault-tolerance, following of stds etc all increase the size for the same functionality

Software…

Student	**Industrial Strength**
Developer is the user - Works for the typical case most of the time - Bugs are tolerable - UI not important - No documentation	Others are the users - Works robustly - Bugs not tolerated - UI very important issue - Documents needed for the user as well as for the organization and the project

Software…

Student	Industrial Strength
SW not in critical use	Supports important functions / business
Reliability, robustness not important	Reliability , robustness are very important
No investment	Heavy investment
Don't care about portability	Portability is a key issue here

Goals of Industrial Strength SE

- **Consistently develop SW with high Q&P for large scale problems, under change**
- Q&P are the basic objectives to be achieved
- Q&P governed by people, processes, and technology

What is software process Model?
What are the Software process activities?

Software engineering is sometimes called a software process. There are four fundamental activities that are common to all software processes for Professional software development.

- Software specification.
- Software development.
- Software validation.
- Software evolution.

SDLC Models will be discussed in Chapter 2.

What is the difference between software engineering and computer science?

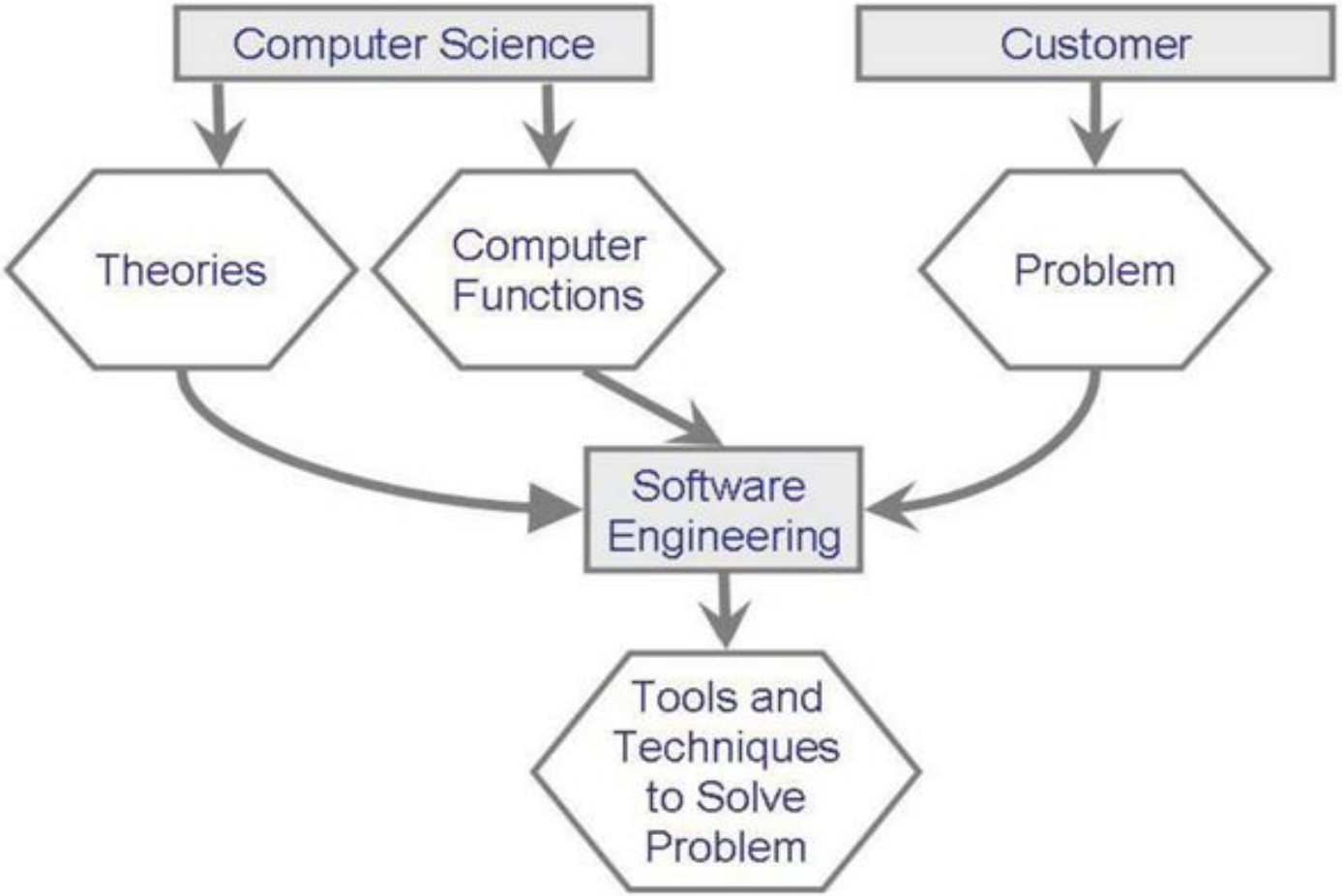
- Computer science is concerned with theory and fundamentals; software engineering is concerned with the practicalities of developing and delivering useful software.
- Computer science theories are still insufficient to act as a complete underpinning for software engineering (unlike e.g. physics and electrical engineering).

Scientist vs Engineer

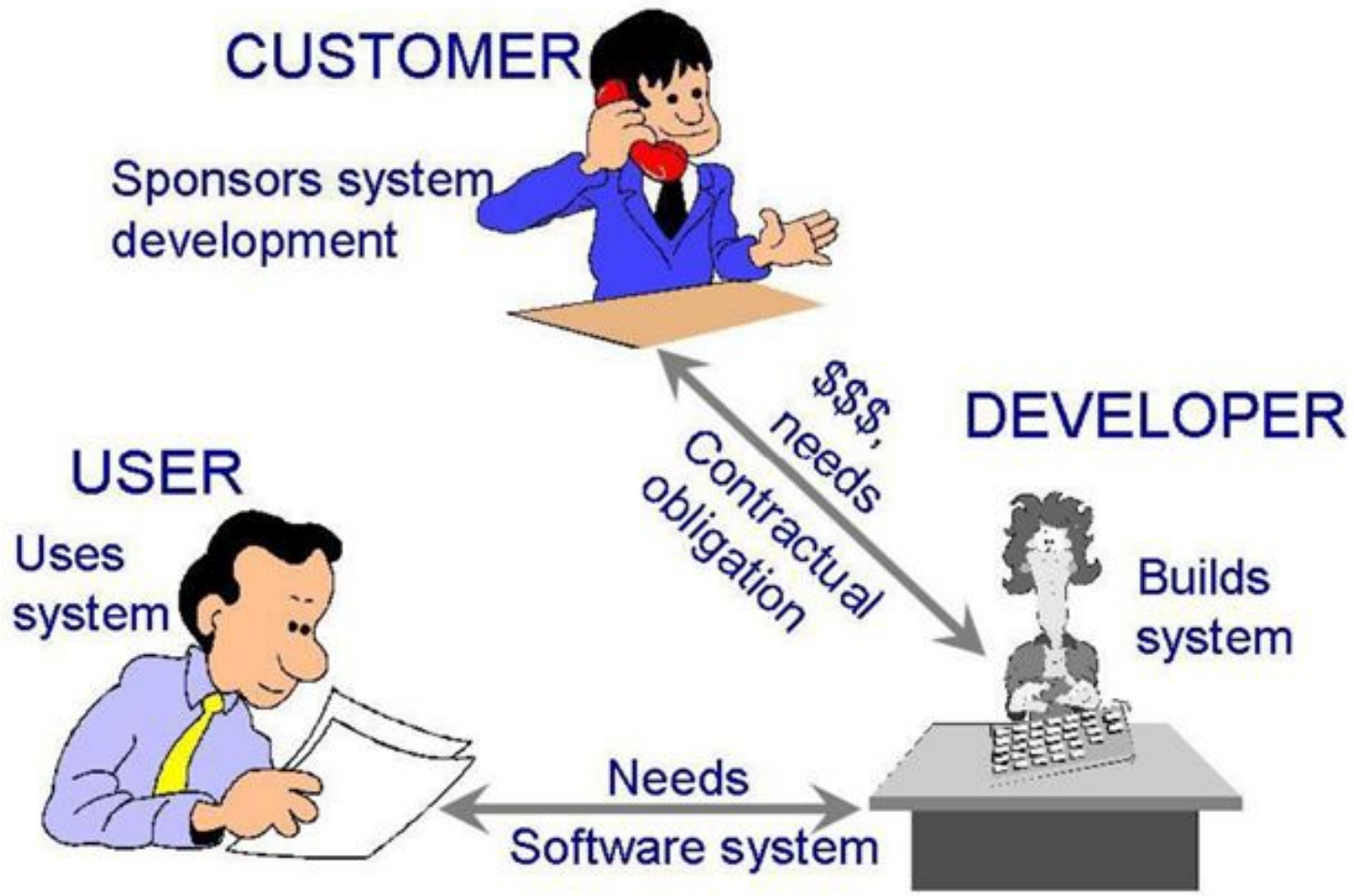
- Computer Scientist
  - Proves theorems about algorithms, designs languages, defines knowledge representation schemes
  - Has infinite time…
- Engineer
  - Develops a solution for an application-specific problem for a client
  - Uses computers & languages, tools, techniques and methods
- Software Engineer
  - Works in multiple application domains
  - Has only 3 months…

Isn't there something more fundamental about "Software" Engineer?

What is the difference between software engineering and computer science?



Who does software engineering?



What are the costs of software engineering?

- Roughly 60% of costs are development costs, 40% are testing costs. For custom software, evolution costs often exceed development costs.
- Costs vary depending on the type of system being developed and the requirements of system attributes such as performance and system reliability.
- Distribution of costs depends on the development model that is used.

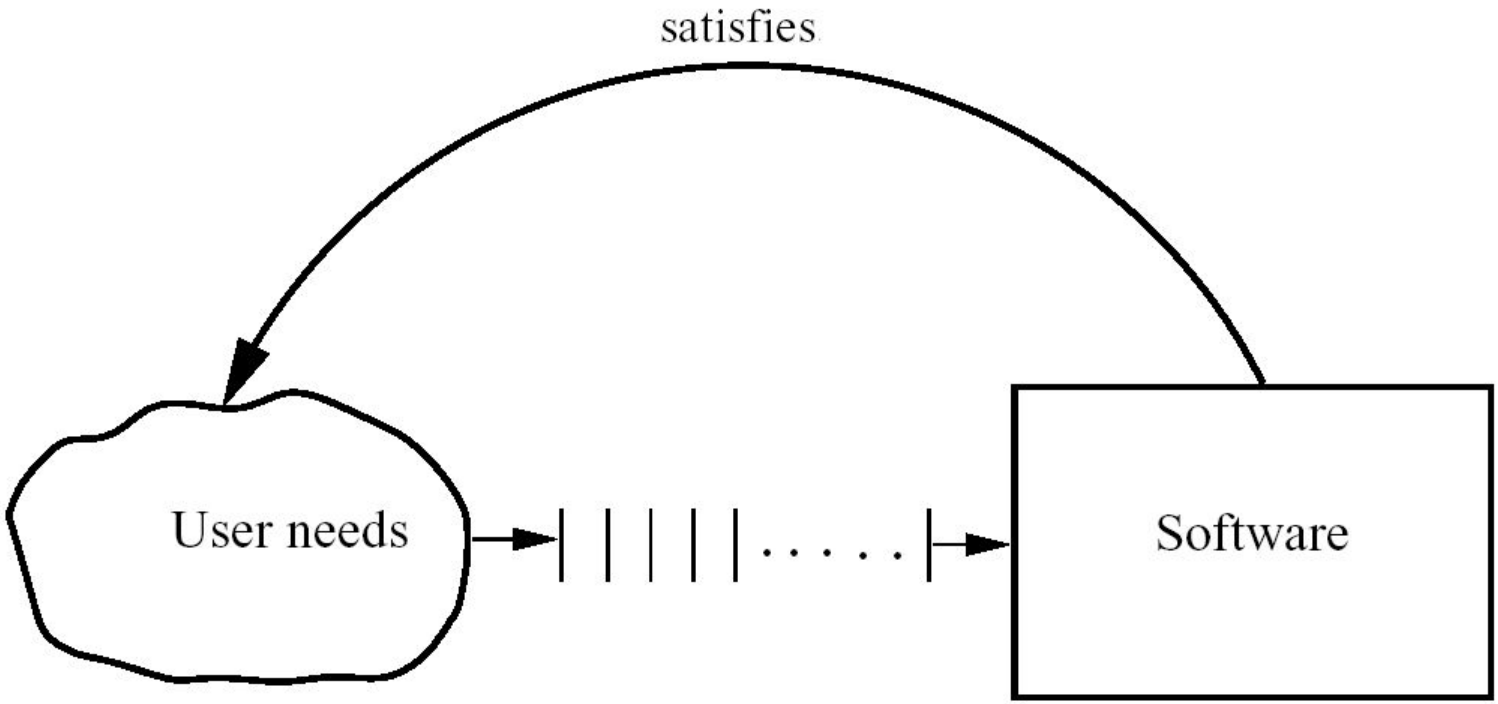
Software is Expensive…

- The HW/SW ratio for a computer system has shown a reversal from the early years.
  - In 50s , HW:SW :: 80:20
  - In 80s , HW:SW :: 20:80
- So, SW is very expensive
  - Importance of optimizing HW is not much
  - More important to optimize SW

Area of Software Engineering?

- Problem domain discussed before, now we discuss the area of SE
- **SE (IEEE): systematic approach to development,…., of software**
- Systematic approach: methodologies and practices that can be used to solve a problem from problem domain

Basic Problem



What are the attributes of good software?

- The software should deliver the required functionality and performance to the user and should be maintainable, dependable and acceptable.
- Maintainability
  Software must evolve to meet changing needs;
- Dependability
  Software must be trustworthy;
- Efficiency
  Software should not make wasteful use of system resources;
- Acceptability
  Software must accepted by the users for which it was designed. This means it must be understandable, usable and compatible with other systems.

SE Challenges

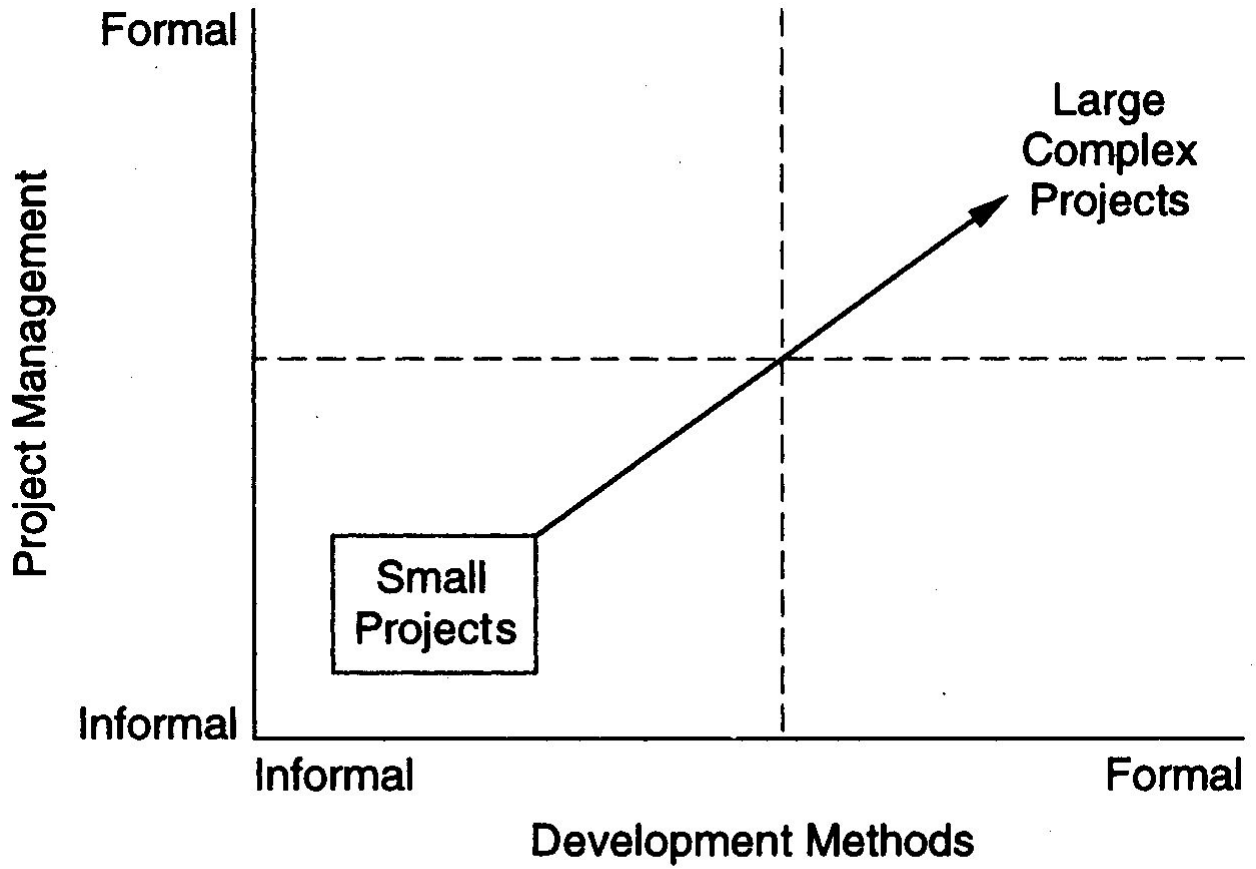
- The problem of producing software to satisfy user needs drives the approaches used in SE

- Q: What other factors that drive the selection of a SE approach?
  1. scale,
  2. productivity,
  3. quality,
  4. consistency,
  5. rate of change, …

1) Scale

- SE must deal with problem of scale
  - methods for solving small problems do not scale up for large problems
  - industrial strength SW problems tend to be large
- SE methods must be scalable
- Two clear dimensions in this
  1. **engineering methods**
  2. **project management**
- For small, both can be informal or ad-hoc; for large, both have to be formalized

1) Scale…


Formal
Project Management
Large
Complex
Projects
Small
Projects
Informal
Informal
Formal
Development Methods

1) Scale…

- An illustration of issue of scale is counting the number of people in a room vs taking a census
  - Both are counting problems
  - Methods used in first not useful for census
  - For large scale counting problem, must use different techniques and models
  - Management will become critical

1) Scale: Examples

Gcc	980KLOC	C, C++, yacc
Perl	320 KLOC	C, perl, sh
Appache	100 KLOC	C, sh
Linux	30,000 KLOC	C, C++
Windows XP	40,000 KLOC	C, C++

2) Productivity

- An engineering project is driven by **cost** and **schedule**
- **Cost:** In sw, cost is mainly manpower cost; hence, it is measured in person-months
- **Schedule** is in months/weeks – very important in business context
- In Biz (Business/Industry) context
  - Cost and Schedule can not be separated
  - SE must serve the Biz, NOT the other way around

Productivity…

- Productivity captures both Cost and Schedule
  - If P is higher, cost is lower
  - If P is higher, time taken can be lesser
- Approaches used by SE must deliver high Productivity

Productivity…

- Q : If you have to write a 10,000 line program in C to solve a problem, how long will it take?
- Answers: generally range from 2-4 months
- Let us analyze the productivity
  - Productivity = output/input resources
  - In SW output is considered as LOC
  - Input resources is effort - person months; overhead cost modeled in rate for person month
  - Though not perfect, some productivity measure is needed, as project has to keep it high

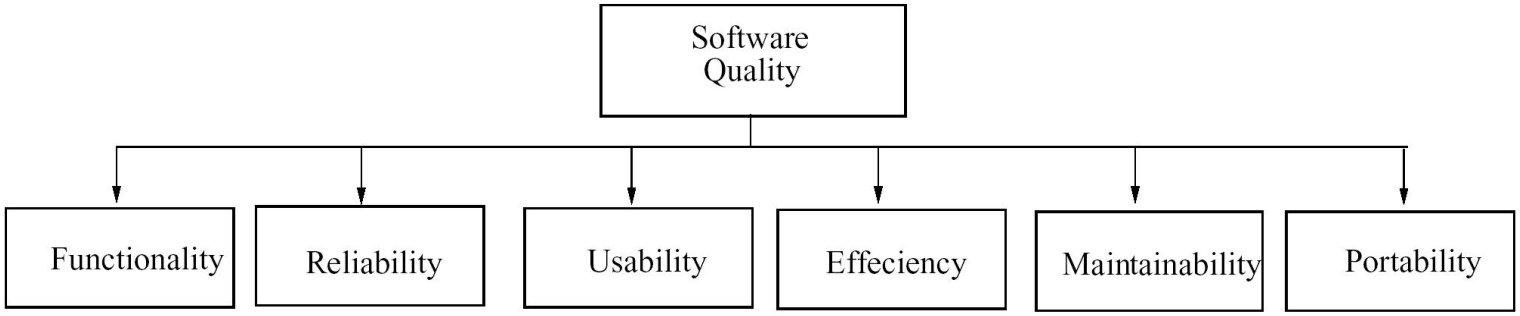
Productivity…

- The productivity is 2.5-5 KLOC/PM
- Q: What is the productivity in a typical commercial SW organization ?
- A: Between 100 to 1000 LOC/PM
- Q: Why is it low, when your productivity is so high? (people like you work in the industry)
- A: What the student is building and what the industry builds are two different things

3) Quality

- Quality is the other major driving factor
- Developing high Quality SW is a basic goal
- Quality of SW is harder to define
- Approaches used should produce a high Quality software

3) Quality – ISO standard



- ISO standard has six attributes
  1. Functionality
  2. Reliability
  3. Usability
  4. Efficiency
  5. Maintainability
  6. Portability

3) Quality…

- Multiple dimensions mean that not easy to reduce Q to a single number
- Concept of Q is project specific
  - For some reliability is most important
  - For others usability may be more important
- Reliability is generally considered the main Q criterion

3) Quality…

- Reliability = Probability of failure
  - Hard to measure
  - Approximated by # of defects in software
- To normalize Quality = Defect density
  - Quality = # of defects delivered / Size
- Defects delivered - approximated with no. of defects found in operation
- Current practices: less than 1 defect/KLOC
- What is a defect? Project specific!

4) Consistency and repeatability

- Sometimes a group can deliver one good software system, but not a second
- **Key SE challenge:** *how to ensure that success can be repeated ?*
- SE wants methods that can consistently produce high Quality SW with high Productivity
- A SW org, wants to deliver high Q&P consistently across projects
- Frameworks like
  - Inter. Org. for Standardization (ISO) and
  - Capability Maturity Model (CMM)
- focus on this aspect

5) Rate of Change

- Only constant in business is change!
- Software must change to support the changing business needs
- SE practices must accommodate change
  - Methods that disallow change, even if high Q and P, are of little value

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.

General issues that affect most software

- Heterogeneity
  - Increasingly, systems are required to operate as distributed systems across networks that include different types of computer and mobile devices.
- Business and social change
  - Business and society are changing incredibly quickly as emerging economies develop and new technologies become available. They need to be able to change their existing software and to rapidly develop new software.
- Security and trust
  - As software is intertwined with all aspects of our lives, it is essential that we can trust that software.

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.

Application types

- Stand-alone applications
  - These are application systems that run on a local computer, such as a PC. They include all necessary functionality and do not need to be connected to a network.
- Interactive transaction-based applications
  - Applications that execute on a remote computer and are accessed by users from their own PCs or terminals. These include web applications such as e-commerce applications.
- Embedded control systems
  - These are software control systems that control and manage hardware devices. Numerically, there are probably more embedded systems than any other type of system.

Application types

- Batch processing systems
  - These are business systems that are designed to process data in large batches. They process large numbers of individual inputs to create corresponding outputs.
- Entertainment systems
  - These are systems that are primarily for personal use and which are intended to entertain the user.
- Systems for modelling and simulation
  - These are systems that are developed by scientists and engineers to model physical processes or situations, which include many, separate, interacting objects.

Application types

- Data collection systems
  - These are systems that collect data from their environment using a set of sensors and send that data to other systems for processing.
- Systems of systems
  - These are systems that are composed of a number of other software systems.

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.

Software engineering and the web

- The Web is now a platform for running application and organizations are increasingly developing web-based systems rather than local systems.
- Web services (discussed in Chapter 19) allow application functionality to be accessed over the web.
- Cloud computing is an approach to the provision of computer services where applications run remotely on the 'cloud'.
  - Users do not buy software buy pay according to use.

Web software engineering

- Software reuse is the dominant approach for constructing web-based systems.
  - When building these systems, you think about how you can assemble them from pre-existing software components and systems.
- Web-based systems should be developed and delivered incrementally.
  - It is now generally recognized that it is impractical to specify all the requirements for such systems in advance.
- User interfaces are constrained by the capabilities of web browsers.
  - Technologies such as AJAX allow rich interfaces to be created within a web browser but are still difficult to use. Web forms with

Web-based software engineering

- Web-based systems are complex distributed systems but the fundamental principles of software engineering discussed previously are as applicable to them as they are to any other types of system.
- The fundamental ideas of software engineering, discussed in the previous section, apply to web-based software in the same way that they apply to other types of software system.

Key points

- 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

- 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.

Chapter 1- Introduction

Lecture 2

Professional and ethical responsibility

- 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 behavior is more than simply upholding the law.

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 organization's sign up to the code of practice when they join.
- The Code contains eight Principles related to the behavior 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.

Code of ethics - principles

- PUBLIC

  Software engineers shall act consistently with the public interest.

- 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.

- PRODUCT

  Software engineers shall ensure that their products and related modifications meet the highest professional standards possible.

Code of ethics - principles

- JUDGMENT
  Software engineers shall maintain integrity and independence in their professional judgment.
- MANAGEMENT
  Software engineering managers and leaders shall subscribe to and promote an ethical approach to the management of software development and maintenance.
- PROFESSION
  Software engineers shall advance the integrity and reputation of the profession consistent with the public interest

Code of ethics - principles

- COLLEAGUES

  Software engineers shall be fair to and supportive of their colleagues

- 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.

Ethical dilemmas

- Disagreement in principle with the policies of senior management.
- Your employer acts in an unethical way and releases a safety-critical system without finishing the testing of the system.
- Participation in the development of military weapons systems or nuclear systems.

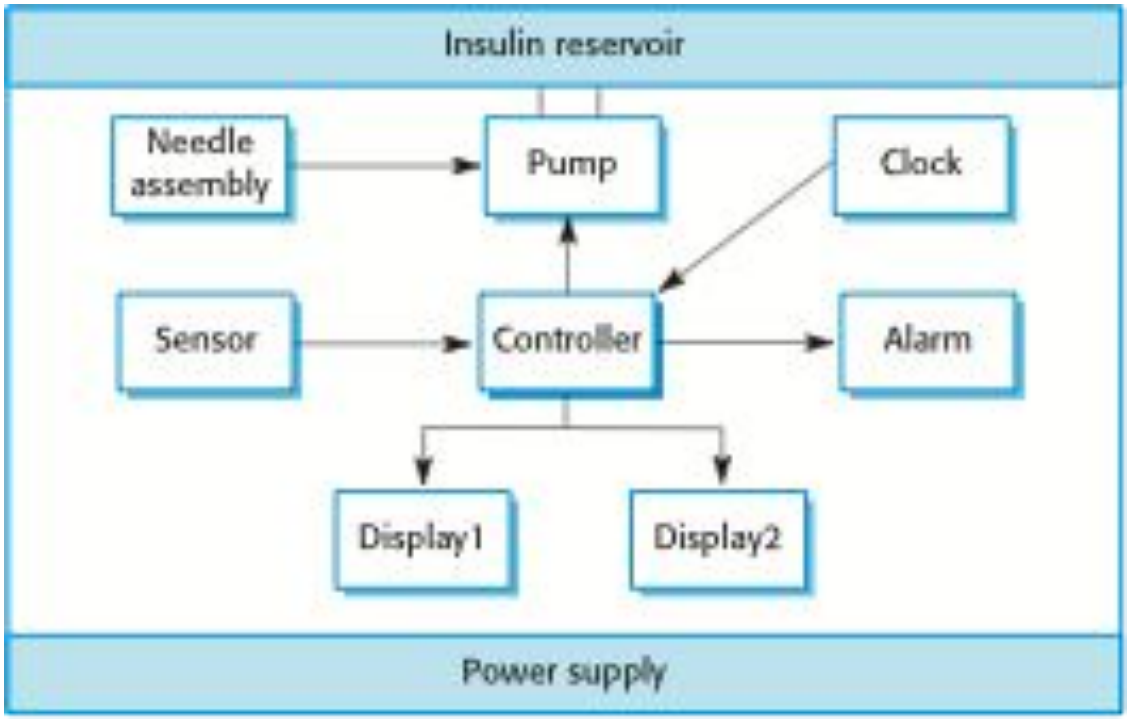
Case studies

- 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.
- A wilderness weather station
  - A data collection system that collects data about weather conditions in remote areas.

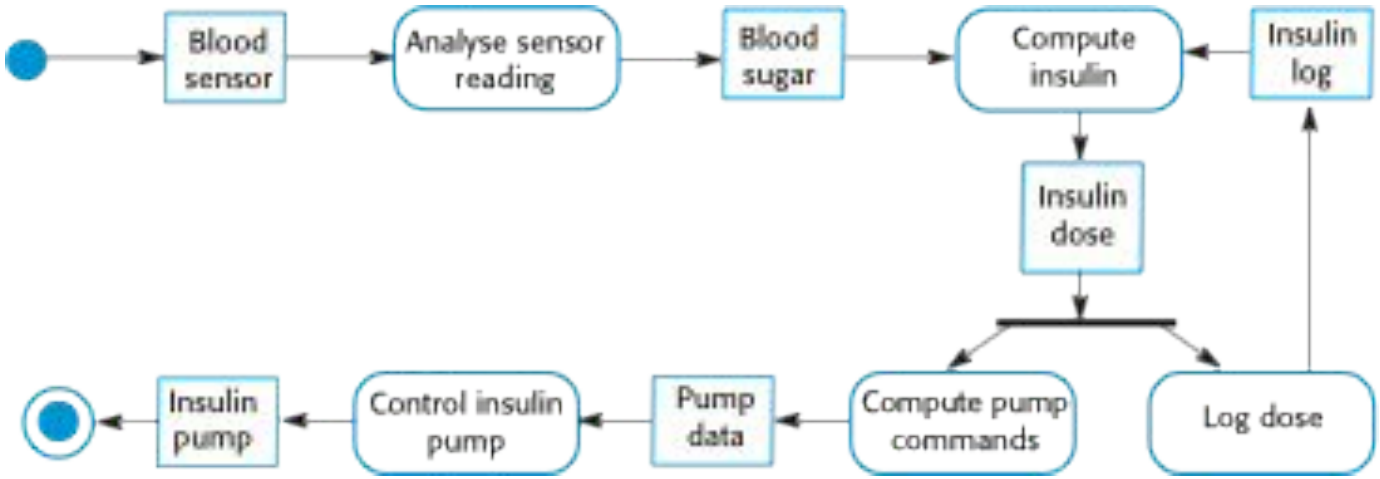
Insulin pump control system

- 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



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.

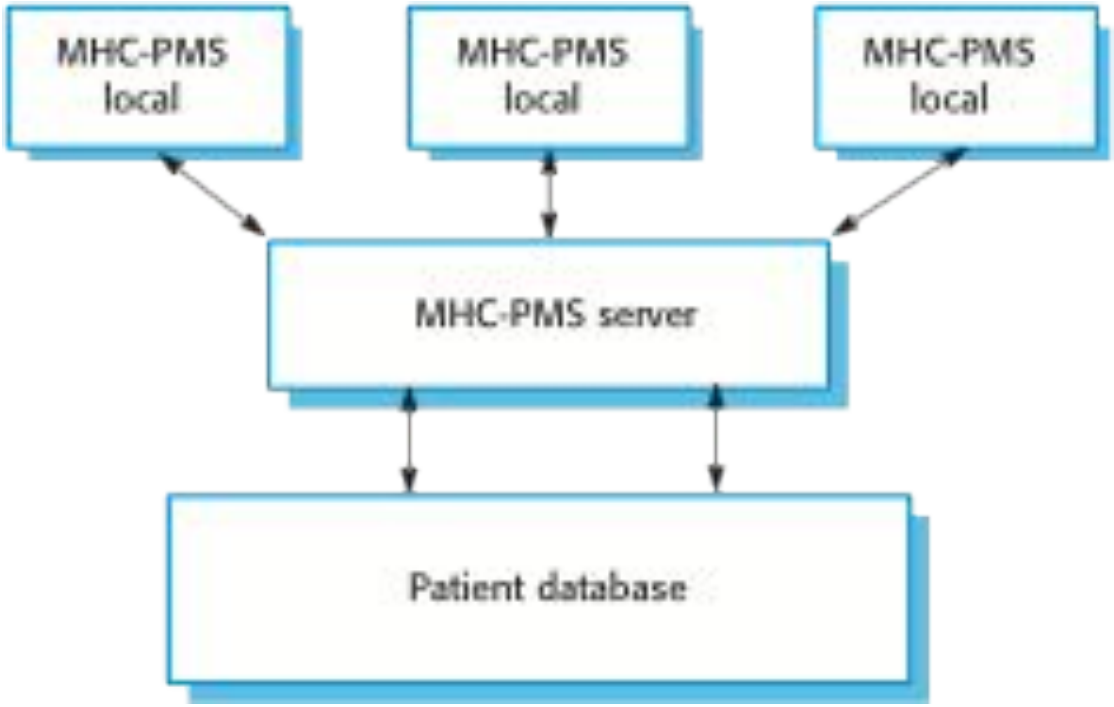
MHC-PMS

- 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



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.

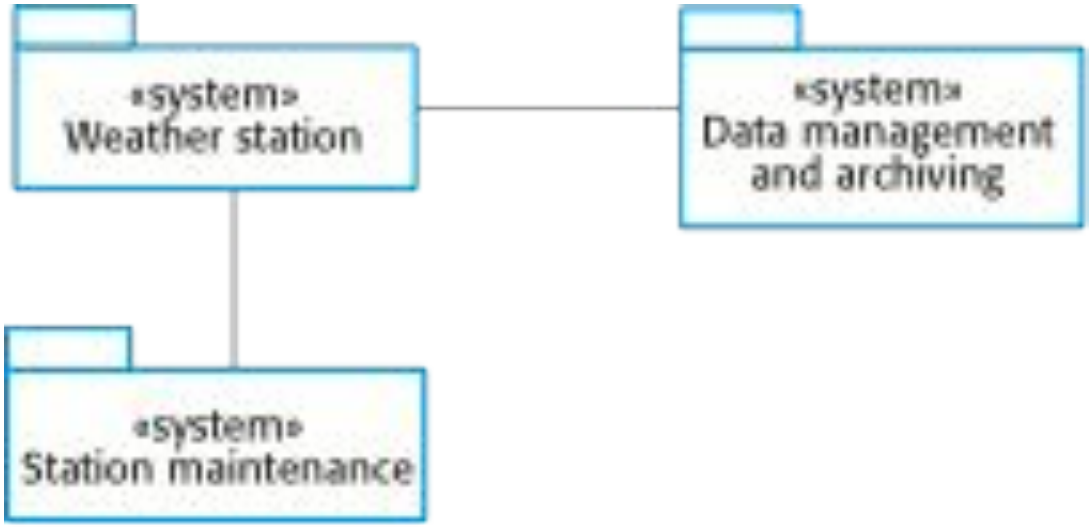
MHC-PMS concerns

- 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.

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.