Software Engineering
CS-303

Introduction

Lecture # 1,2 3
21, 22, 23 Jan

Rubab Jaffar
rubab.jaffar@nu.edu.pk

Today's Outline

- Administrative Stuff
- Overview of 303
- What software is?
- What is engineering approach?

About me

- Graduated BS(SE) from FJWU, RWP
- Complete Masters MS(SE) from NUST, ISB
- Research interests:
  - Software engineering
  - Object Oriented Design
  - Database systems
  - Algorithm analysis

Administrative Stuff

Office hours

- Office 6 (CS building)
- Office hours: 3:00 to 4:00 pm
  - (Tuesday, Wednesday, Thursday)

About the course

- Study and application of the principles and techniques of computer science, engineering, and mathematical analysis to the design, development, testing, and evaluation of the software and the systems that enable computers to perform their many applications.

- This is perhaps the most important course in your CS curriculum in CAREER PATH FORECAST.

- Various cases studies will be used throughout the course to demonstrate the concepts learnt.

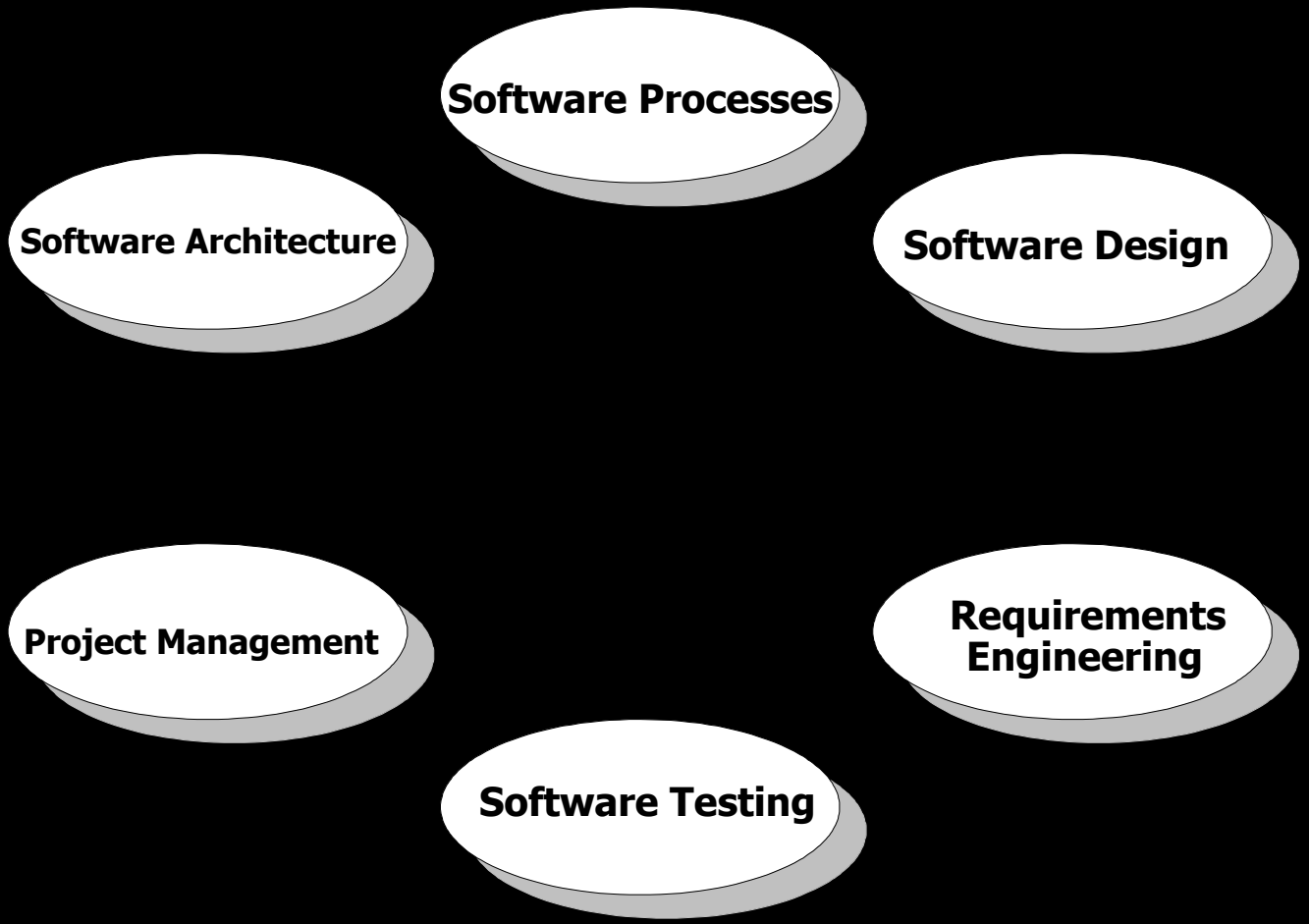
- A strong in class participation from the students will be encouraged and required during the discussion on these case studies.

Career Forecast

- Software engineers are in demand in not only at software development companies but also in all other organizations that are involved in the development of significant information systems – including
  - governments,
  - telecommunications companies,
  - the chemical industry,
  - the bio-medical industry,
  - financial institutions,
  - pharmaceuticals,
  - healthcare sector corporations,
  - engineering and manufacturing firms,
  - e-commerce,

In fact in the words of noted software engineer, David Parnas, "career opportunities for software engineers are essentially unlimited."

Major Topics of the course



Course Outline

- Introduction to Software Engineering
- Software Processes
- Agile Development
- Software Requirements engineering processes
- System models
- Architectural design
- Application architectures
- User interface design
- Software testing, Quality management
- Project Management
- Managing people
- Risk Management
- Software cost estimation

Pre-requisites /Knowledge assumed

- Object Oriented Analysis and Design
- Programming language concepts
- Data structure concepts
- Database system concepts

Skills assumed

- We assume you have the skills to code in any programming language therefore you
  - can design, implement, test, debug, read, understand and document the programs.

Tentative Mark Distribution and Grading Policy

- Assignments/Class Participation: 10 %
- Quizzes: 10%
- Mid Exams: 30 %
- Final: 40%
- Presentations + Projects: 10%
- Absolute Grading Scheme

Course Ethics

Projects/Assignments

- Deadlines are always final
- No credit for late submissions
- One student per assignment at maximum
- Project Proposal Submission : 3rd Week
- Project Presentation : Last week

Quizzes

- Announced
- Unannounced

Honesty

- All parties involved in any kind of cheating in any exam will get zero in that exam.

Project

- An advance project
  - Provides interesting problem, realistic pressures, unclear/changing
- 3 member teams
- Emphasis on good SE practice/methodology
  - Homework's and deliverables tied to the project

Course Material

- You will have Presentations of each topic and ***reference books*** in PDF format will be available on slate.

Text Books

1. Ian Sommerville, Software Engineering 10th Edition

Reference Books

1. Pressman, R S Software Engineering: A Practitioners Approach (7th Edition, European Adaptation), McGraw Hill, 1994

Course Goals/Objectives

- By the end of this course, you will
  - Have a sound understanding of the fundamental concepts of the software engineering paradigm
  - Understand and apply different practices used in software industry for software development
  - Recognize the risks of software failure
  - Verify through review practice accuracy, ambiguity and completeness of a software
  - Become familiar with different tools used by industry in the software development process
  - Learn professional practices to apply in future
  - Develop a software engineering mindset

Why study Software Engineering ?

Software Products

- Generic products
  - Stand-alone systems that are marketed and sold to any customer who wishes to buy them.
  - Examples – Word Processors, graphics programs, Information systems, ERPs
- Customized products
  - Software that is commissioned by a specific customer to meet their own needs.
  - Examples – embedded control systems, air traffic control software, NADRA software

Product Specification

- Generic products
  - 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
  - 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.

Lec 2: Today's Outline

- What Engineering Approach is?
- How Software Engineering is different?
- Software Engineering Methods
- Professional and Ethical Issues in SE

Engineering Approach

Before discussing this we need to understand difference between **science** and **Engineering**

Difference b/w Science & Engg

- **What is Science ?**
  - Scientist invented a ***wheel***,
  - ***Engineer*** will use that wheel for bi-cycle, motor-car, for anything.

  - Scientist invented an ***electric motor***,
  - ***Engineer*** will use that motor to construct things like pumps, electric-lifts, toys, etc.

- **What is Engineering ?**
  - Process of ***productive use*** of scientific knowledge is called engineering.

Difference b/w CS & SW-Engineering

- ***Computer science*** is knowledge contains theories and fundamentals.
- Somebody has invented followings
  - Database
  - Design methodology
  - Algorithms
  - Testing methods
  - Methods of analysis and Verification.

- **Software Engineering** is concerned with the practicalities of developing and delivering useful software.
- **SW-Engineering** follows engineering which is to plan, with suitable skills and craft for moving into Mass-production.

Engineering Projects

- Day to day life we see successful projects in other fields of engineering
  - Such as in civil, or mechanical engineering.
  - Building bridges & roads are also large projects and the carryout large efforts in construction.

- It is our common observation that usually large software projects are NOT very successful.
  - In the middle of software projects we realize that **more time** is required or
  - **more resources** are required to complete the project, than we initially anticipated.

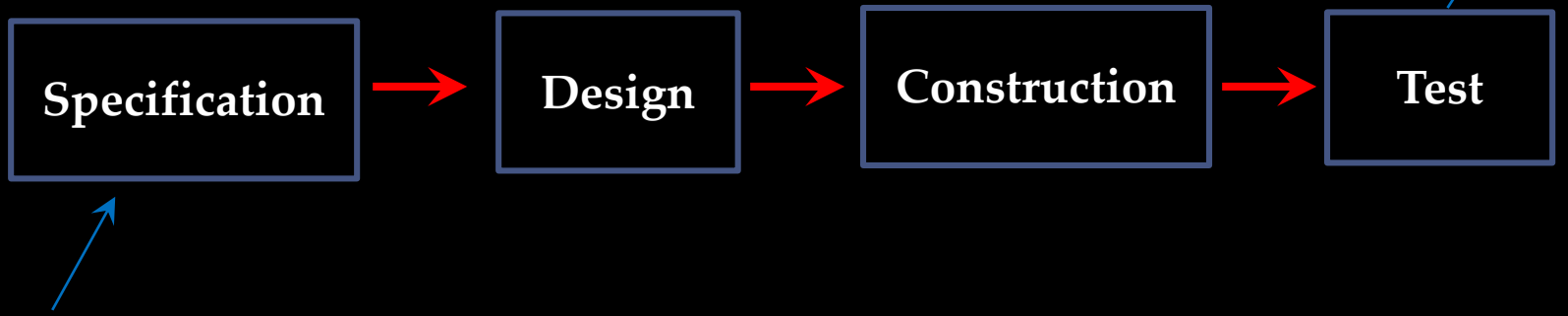
Engineering Approach

- Following are ***Large*** engineering projects ***in other discipline*** and are normally successful
  - Building bridges & roads
  - Power Plants
  - 60 story luxury apartments
  - Aircraft missiles
- Why such kinds of project are generally successful?
- What are the ***practices being followed*** in these kinds of project that make them successful?
- Is there any ***common attributes*** or these are different than developing a software?

Engineering Approach

Using this approach →

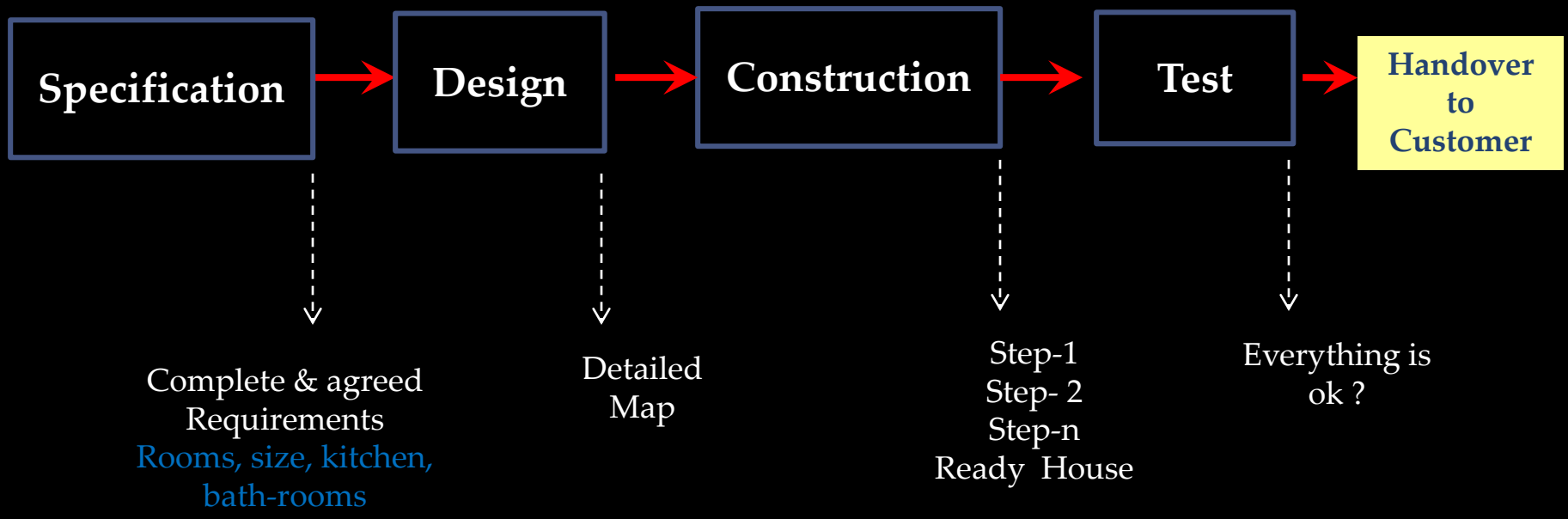
- Construct a house
- Construct a bridge
- Or design a CAR



This ***engineering approach*** helps engineers developing what they intend to and complete projects successfully

Example – Construct a house

How Engineering approach helped **construction** of **House** ?



Software Engg is Different

Software Engineering is different

- Requirements are complex and ***multi-dimensional.***
- Difficult to asses ***progress*** of the project.
- Delivering an ***intangible*** product.
- Software ***get obsolete*** quickly.
- Can be developed using ***different approach***
  - Approach = process, templates, techniques etc
  - Plan-driven (Predictive) & change-driven (adaptive)

Requirements are complex

Components of any automation Solution

- Features and functions
- Process/Practices (need to be automated)
- People (their experience, knowledge, and ability to adopt change)
- Usability (Desktop, Internet, Smart-phone)
- Environment (Temp, Dust-free etc)

Guess my requirements

I need an equipment

- That is equipped with internal combustion engine (such as Car, truck)
- That is available in yellow & black colour
- It has three wheels
- Has 3 speed forward and 3 speed reverse
- Levels uneven surface nicely and quickly

Difficult to asses progress

- You can asses progress in civil engineering
  - **50%** work is completed (in 60 story building)
- When Project Manager of software project says 90% work complete.
  - How can you verify that statement **?**
  - Is it really 90% or much less . . .

Delivering an intangible thing

- In civil engineering you are delivering house or a road or a bridge.
  - Can understand the time taken and cost
  - Buy a Car you look for rooms, their size, design etc

- In software project what do you deliver ?
- A CD only cannot asses ***how much efforts*** it had taken and ***what is its cost***.

Can differentiate - 2 kilo potato or 20 Kilos
Cannot differentiate - Small SW or large SW

Software get obsolete quickly

- Software is developed to ***automate business process***, as business grows its process changes.
- Need some changes approximately after very 8-9 months in developed software.
- This is ***not that case*** when you purchased a ***brand-new car*** or new house.

Can be developed using different approach

- Software house use different approaches to develop same functionality
  - Plan-Driven (Sequential or Predictive)
  - Change-Driven (Agile or Adaptive)

Software Engineering is Different

- It is difficult to estimates efforts and plan these activities for software development.
- There are so many failures in software development projects.
- Typical engineering approach is necessary for the successful software project.
- We must have, separate specialized team for . . .
  - Requirements elicitation
  - Dedicated team for software design
  - Team of developers
  - separate team for testing etc.

Why study Software Engineering ?

- What are ***challenges*** in software development ?
  - that will be addressed by using Engineering approach

- What is the need/required ?
  - Unable to handle large software projects - Why
- Why are we studying software Engineering?

Why Study Software Engineering ?

- Historically 2-3 people used to develop ***small*** and ***simple*** software that is ok. No problems in small & medium size projects
  - Example: your FYP is small and simple project

- When a ***large*** software with ***complex*** requirements are developed through a large team (70-80 people) we had serious problems (called *Software crisis*).
  - Example: Developing a core banking software is large project, will require large team and few years.

Software Crisis

- In early ***60s*** software techniques that were used to develop ***small*** software were ***not applicable*** for ***large*** software applications.
  - In most of the cases that software which was tried to be build using those old tools and techniques were either ***not complete***.
  - Most of the times it was delivered ***too late***.
  - Most of the projects were ***over-budgeted***.
  - A few projects caused loss of life
  - Some projects caused property damage.

Cost of Failure

- ***Allstate Insurance*** – (www.allstate.com) in 1982
  - **$8M** were allocated to automate business
  - EDS providing software
  - Initial *5-year* project continued for *10-years*, until 1993
  - Cost approached $100M

- ***Bank of America*** – (www.bankofamerica.com)
  - Spent **$23M** on an initial **5 year** accounting & reporting system
  - Spent $600M trying to make it work
  - Project was *cancelled*
  - Lost customer accounts - $Billons

Cost of Failure

- ***Blue Cross and Blue Shield of Wisconsin*** – 1983
  - EDS was hired to build ***$200M*** computer system
  - Delivered on time in 18 months
  - System didn't work – issued $60M in overpayments and duplicate checks
  - Blue Cross lost 35,000 policy holders by 1987

NATO Conference

- In the fall of 1968 and again the in fall of 1969, NATO hosted a conference devoted to the subject of software engineering.

- The motivation for these conferences was that the computer industry at large was having a great deal of trouble in producing large and complex software systems i.e. the software crisis.

- Participants were solicited from computer manufacturers, computer users, software houses, and academia. Over the years, the conference reports have gained a certain amount of classical aura.

NATO Conference

- In late 1968 the Conference recommended the term Software Engineering. The phrase 'software engineering' was deliberately chosen as being provocative, in implying the need for software manufacture to be based on the types of theoretical foundations and practical disciplines, that are traditional in the established branches of engineering.

Lec 3: Topics covered

- 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.

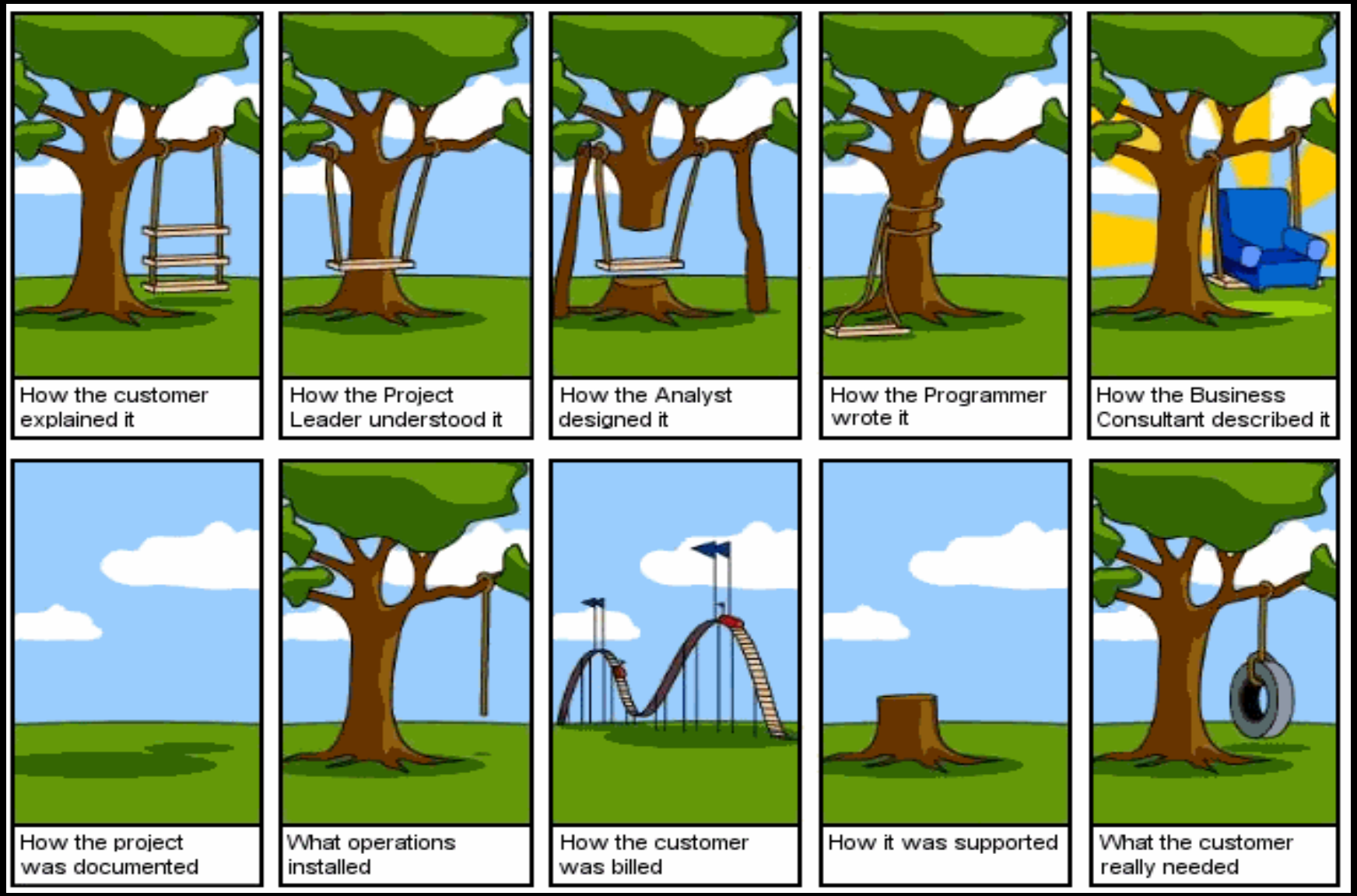
Software Engineering

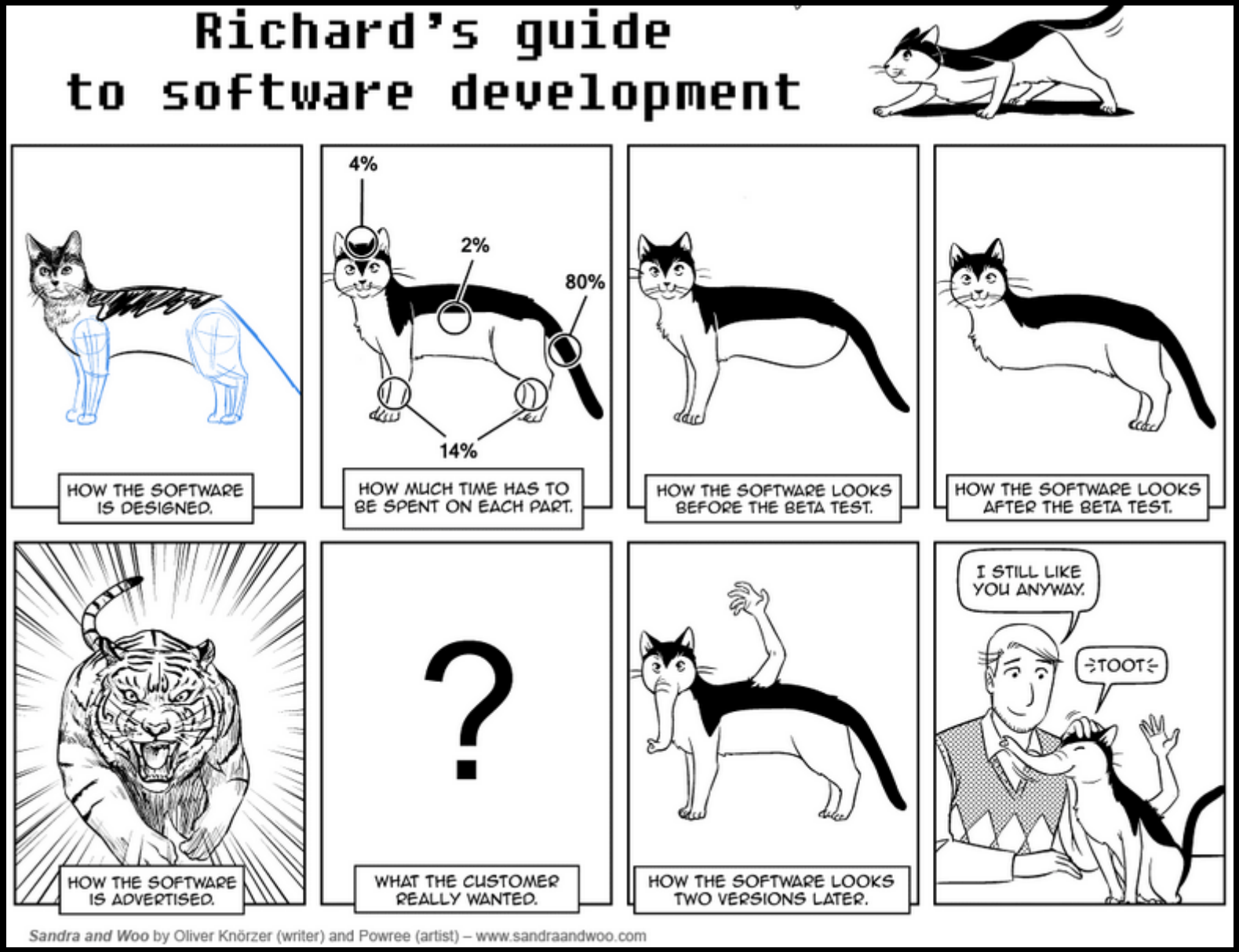
- Writing software has evolved into a profession concerned with
  - How best to maximize the quality of software and of how to create it.
  - How best to create high quality software is a separate and controversial problem covering software design principles
  - How best to deliver software on time and as quickly as possible, work-place "culture", hiring practices, and so forth.

All this falls under the broad rubric of software engineering


www.nytimes.com/2015/08/11/opinion/bina-shah-pakistan-the-next-software-hub.html
HOME
SEARCH
The New York Times
The Opinion Pages | CONTRIBUTING OP-ED WRITER
Pakistan, the Next Software Hub?
AUG. 10, 2015
Pakistan isn't usually considered one of the world's information technology powerhouses; its share of global I.T. sales is only $2.8 billion, of which $1.6 billion represents tech and I.T. services and software exported abroad. This is a tiny percentage of the expected $3.2 trillion global market for 2015, and is dwarfed by India's $100 billion worth of software exports per year.

Why SE?


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


Richard's guide to software development
HOW THE SOFTWARE IS DESIGNED.
4%
2%
80%
14%
HOW MUCH TIME HAS TO BE SPENT ON EACH PART.
HOW THE SOFTWARE LOOKS BEFORE THE BETA TEST.
HOW THE SOFTWARE LOOKS AFTER THE BETA TEST.
HOW THE SOFTWARE IS ADVERTISED.
?
WHAT THE CUSTOMER REALLY WANTED.
HOW THE SOFTWARE LOOKS TWO VERSIONS LATER.
I STILL LIKE YOU ANYWAY.
÷TOOT÷
Sandra and Woo by Oliver Knörzer (writer) and Powree (artist) – www.sandraandwoo.com

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 process concerned with developing the software infrastructure, control, applications and databases in the system.

Attributes of good software

- The software should deliver the required functionality and performance to the user and should be
  - Maintainable,
  - Dependable (Reliable, safe, secure)
  - Efficient (memory, process cycles)
  - Acceptable (understandable, usable and compatible with other systems)

Software Engineering

- The economies of ALL developed nations are dependent on software.
- More and more systems are software controlled
- Software engineering is concerned with theories, methods and tools for professional software development.
- Expenditure on software represents a significant fraction of GNP in all developed countries.

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 Project Failure

- *Increasing system complexity*
  - As new software engineering techniques help us to build larger, more complex systems, the demands change. Systems have to be built and delivered more quickly; larger, even more complex systems are required; systems have to have new capabilities that were previously thought to be impossible.
- *Failure to use software engineering methods*
  - It is fairly easy to write computer programs without using software engineering methods and techniques. Many companies have drifted into software development as their products and services have evolved. They do not use software engineering methods in their everyday work. Consequently, their software is often more expensive and less reliable than it should be.

Professional Software Development

Software Process Activities

- Software specification,
- Software development,
- Software validation,
- Software evolution,

General issues that affect 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.
- Scale
  - Software has to be developed across a very wide range of scales, from very small embedded systems in portable or wearable devices through to Internet-scale, cloud-based systems that serve a global community.

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 modeling 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.
- Safety Critical System
  - A safety-critical system (SCS) or life-critical system is a system whose failure or malfunction may result in one (or more) of the following outcomes: death or serious injury to people. loss or severe damage to equipment/property. environmental harm.

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.

Internet Software Engineering

- 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-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 apply to web-based software in the same way that they apply to other types of software system.

Web Software Engineering

- Software reuse
  - 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.
- Incremental and agile development
  - 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.
- Service-oriented systems
  - Software may be implemented using service-oriented software engineering, where the software components are stand-alone web services.
- Rich interfaces
  - Interface development technologies such as AJAX and HTML5 have emerged that support the creation of rich interfaces within a web browser.

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.

Rationale for the code of ethics

- *Computers have a central and growing role in commerce, industry, government, medicine, education, entertainment and society at large. Software engineers are those who contribute by direct participation or by teaching, to the analysis, specification, design, development, certification, maintenance and testing of software systems.*
- *Because of their roles in developing software systems, software engineers have significant opportunities to do good or cause harm, to enable others to do good or cause harm, or to influence others to do good or cause harm. To ensure, as much as possible, that their efforts will be used for good, software engineers must commit themselves to making software engineering a beneficial and respected 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.

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

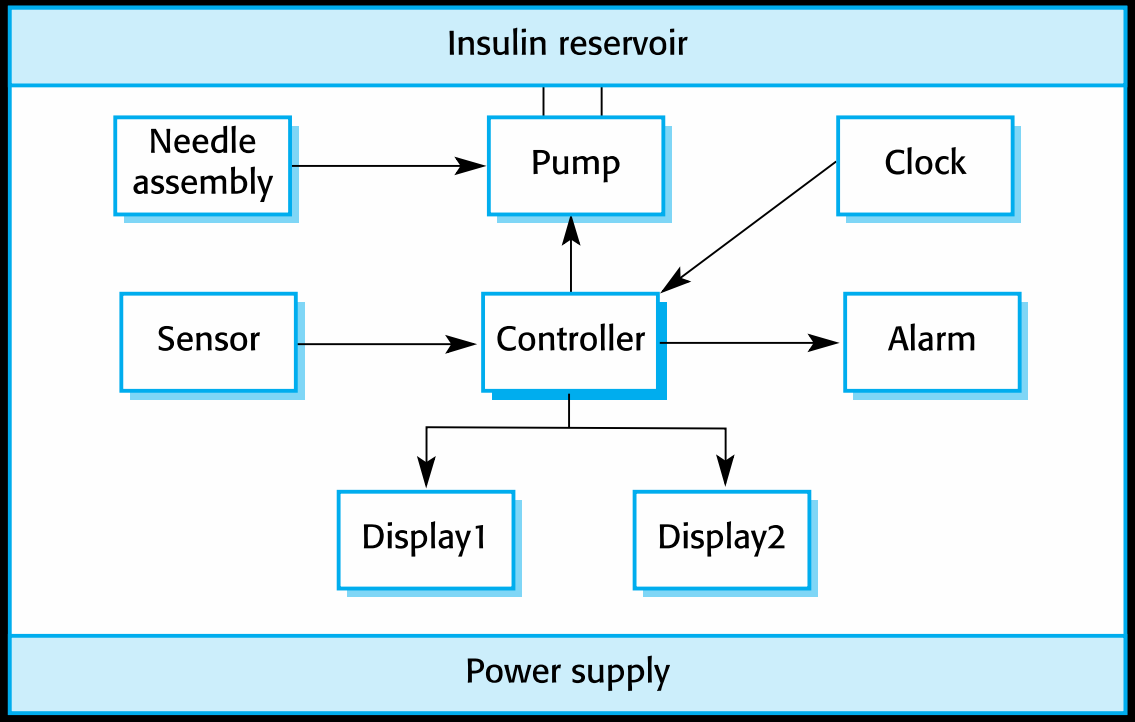
Case studies

- 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
  - Mentcare. 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.

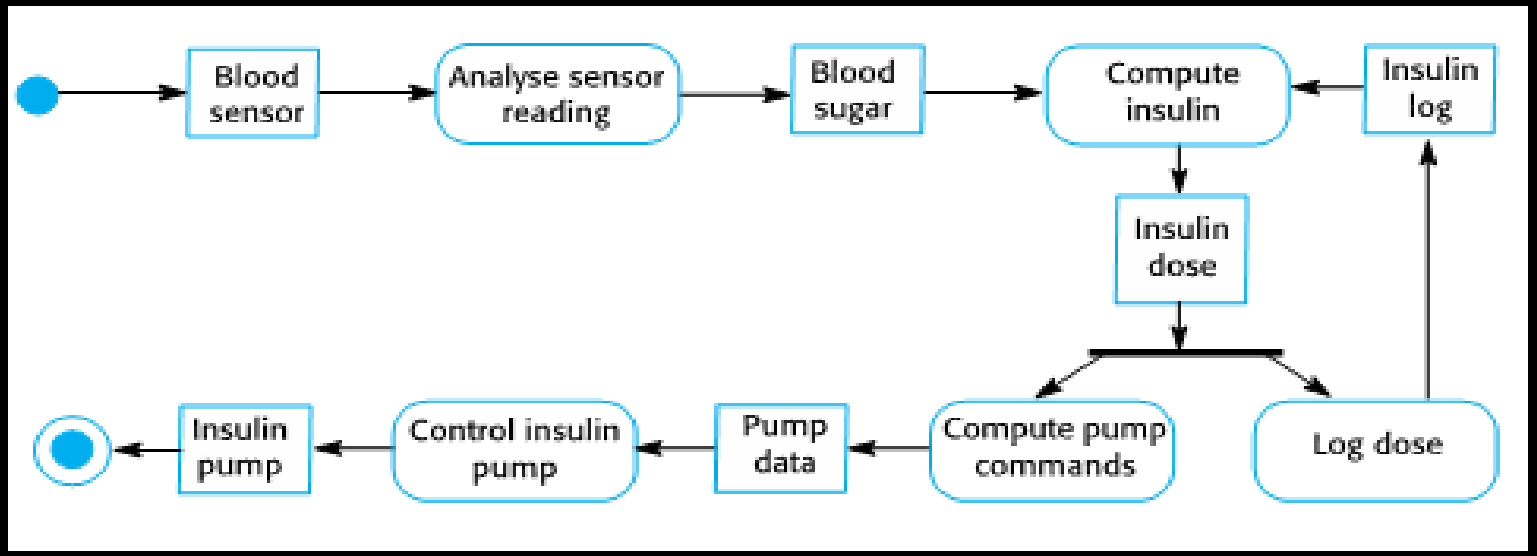
Insulin pump control system

- 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.

Mentcare: 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.

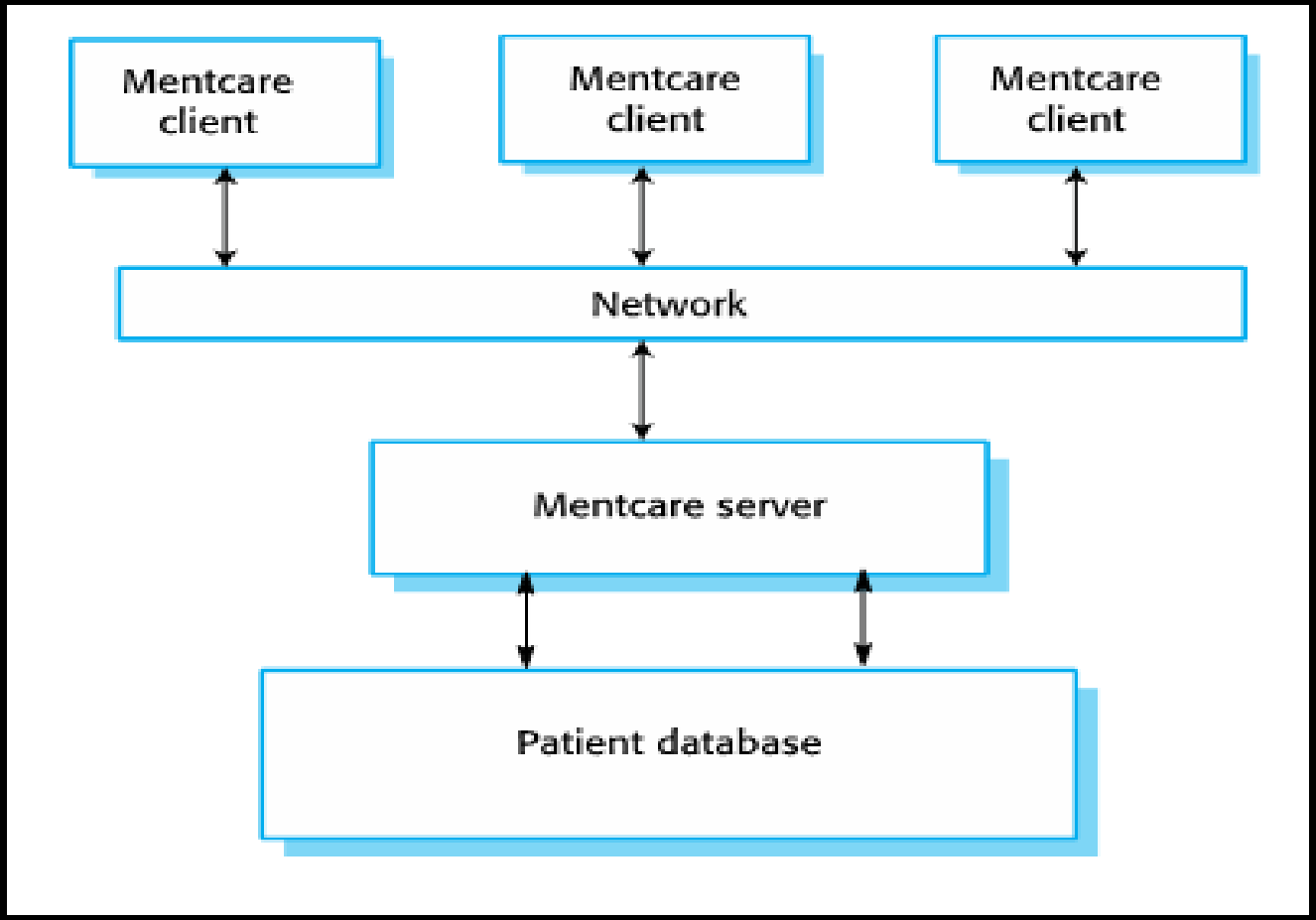
Mentcare

- Mentcare 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.

Mentcare 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 Mentcare System



Key Features of the Mentcare System

- 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.

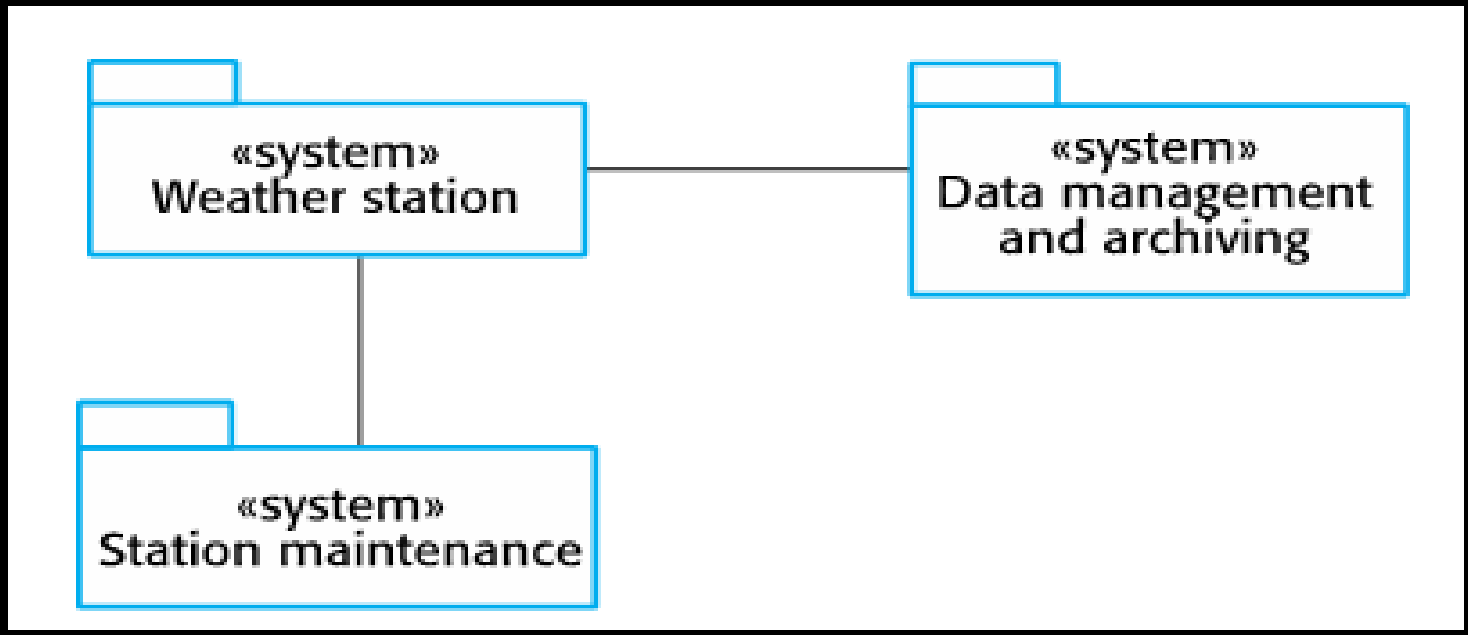
Mentcare System Concerns

- 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.

iLearn: A digital learning environment

- A digital learning environment is a framework in which a set of general-purpose and specially designed tools for learning may be embedded plus a set of applications that are geared to the needs of the learners using the system.
- The tools included in each version of the environment are chosen by teachers and learners to suit their specific needs.
  - These can be general applications such as spreadsheets, learning management applications such as a Virtual Learning Environment (VLE) to manage homework submission and assessment, games and simulations.

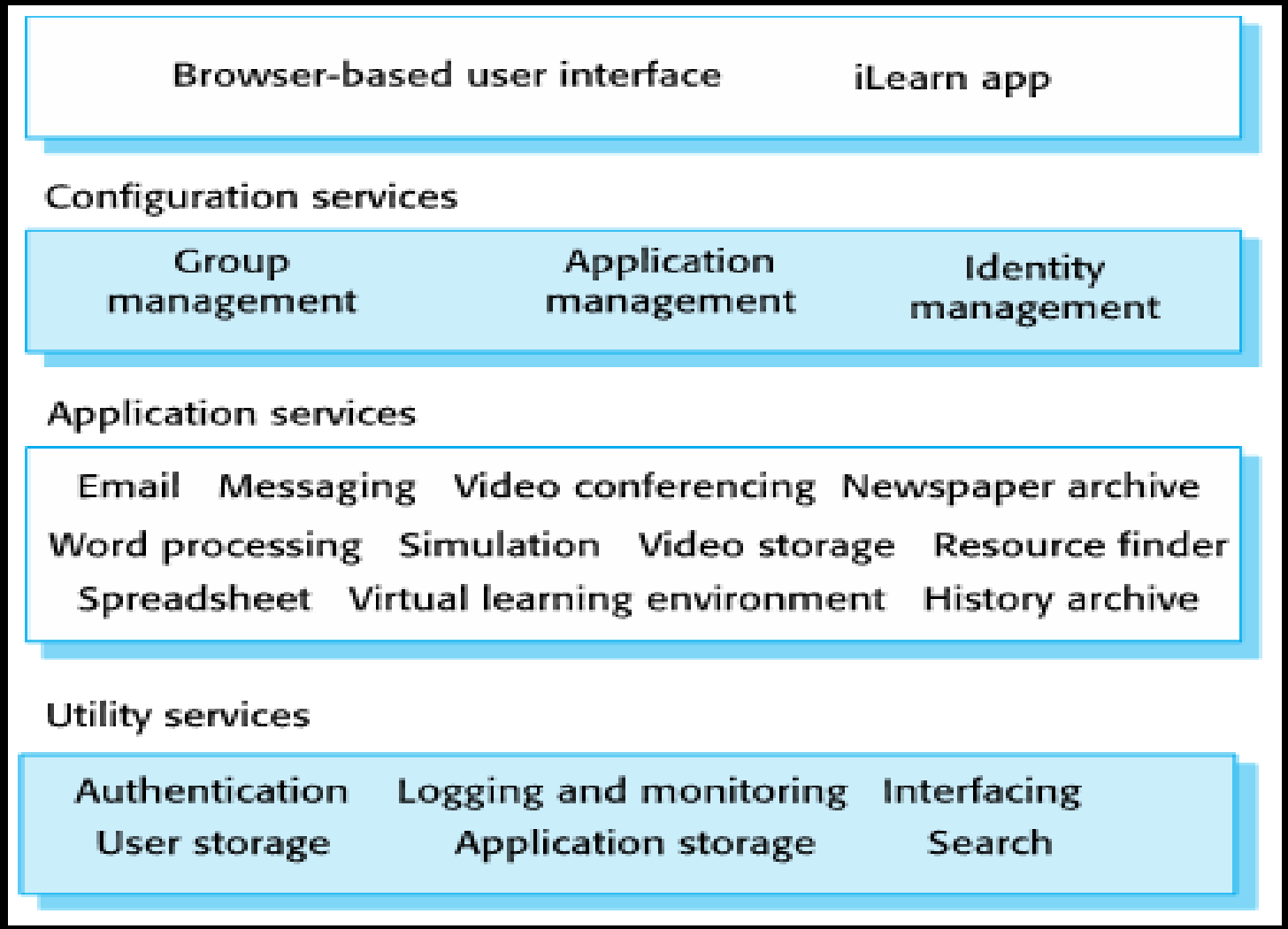
Service-oriented systems

- The system is a service-oriented system with all system components considered to be a replaceable service.
- This allows the system to be updated incrementally as new services become available.
- It also makes it possible to rapidly configure the system to create versions of the environment for different groups such as very young children who cannot read, senior students, etc.

iLearn Services

- *Utility services* that provide basic application-independent functionality and which may be used by other services in the system.
- *Application services* that provide specific applications such as email, conferencing, photo sharing etc. and access to specific educational content such as scientific films or historical resources.
- *Configuration services* that are used to adapt the environment with a specific set of application services and do define how services are shared between students, teachers and their parents.

iLearn Architecture



iLearn Service Integration

- *Integrated services* are services which offer an API (application programming interface) and which can be accessed by other services through that API. Direct service-to-service communication is therefore possible.
- *Independent services* are services which are simply accessed through a browser interface and which operate independently of other services. Information can only be shared with other services through explicit user actions such as copy and paste; re-authentication may be required for each independent service.

That is all