Unit 1: Introduction to information Systems

Data information and knowledge

- **Data**: data is the raw facts and figures. Data by and of itself is meaningless without context.
- **Information**: data that is given context, or changed into a meaningful form becomes information. Data + Meaning = Information
- **Knowledge**: knowledge is produced as a result of understanding information and be used to solve problems.

For example, the number sequence 1,1,2,3,5,8,13 out of context is meaningless. However, with the context that these numbers for the start of the Fibonacci sequence, the data becomes information.

Types of knowledge

Explicit knowledge is knowledge that can easily be passed on to others, such as a person's date of birth.

Tacit knowledge is knowledge that is hard to pass on to others. An example of tacit knowledge would be someone's reflex skills as a goal keeper.

Information Systems

Firstly, it is important to understand that information itself transcends technology and that technology can merely be used as a mechanism to store information.

An information system ties together information technology, data and people to drive business requirements.

Information's systems are composed of 5 core components:

- Hardware
- Software
- Data
- Processes
- People

Many everyday services rely upon meaningful information and knowledge from data. Services such as online shopping and social networks heavily rely upon information systems to operate. The raw data stored in a database can be operated upon to make predictions.

What happens when information systems go wrong?

Problems in information systems can lead to wrong decisions to being made. For example, a glitch in a prisoner's release information system could (and has) lead to prisoners being released early.

Further if one information system reports incorrectly, the damage can be magnified if this information is fed into other information systems “downstream”.

System development lifecycle

The SDLC provides a framework for creating systems that are useful, functional and well tested.

The SDLC is structured in 7 phases:

- Planning
- Analysis
- Design
- Development
- Testing
- Deployment
- Maintenance

1. Planning – During the planning phase we should try to understand the problem, determine solutions and in doing so understand the scope of the work. At this phase, costing and time scales for delivery should be produced. The strengths weaknesses of the system should also be considered.
2. Analysis- During this phase a set of requirements should be captured that detail the requirements of the system. Each requirement is a statement of intent of what the service will offer. A requirement does not need to detail how it would be implemented. There are two categories of requirement: functional (services the system should provide) and non-functional, requirements that can be used to judge a systems correct operation. There are also domain requirements, these are requirements that are specific to the domain specific to the system. This may include things like specific interfaces in the domain.
3. Design – A design is created during this phase that satisfies all the requirements. The design should make use of best practice and established patterns. A good design should. It is important that the design should not modify any of the requirements captured during analysis.
4. Development- During this phase the design in implemented. Let the coding begin!
5. Testing- The implementation must now be tested to provide confidence. There are multiple types of tests that can be run including, unit testing, integration testing, performance testing and security testing.
6. Verification and validation - During this phase we ensure the system meets its requirements.
7. Deployment – This is the Proces of deploying the system into the “real world”.
8. Maintenace – After deployment the system is used by real people. Maintenace involves implementing feature request and fixes bugs. This will be achieved by filtering change requests into the SDLC.

Software Engineering

Computer science vs software engineering

Computer science is concerned with the theory that underlies computers while software engineering is the discipline of building software. While software engineering focusses on producing software, system engineering focuses on specifying the architecture of the whole system, including software and hardware.

Software engineering process

Software engineering (The book by Ian Sommerville) describes a software process as a set of activity required to produce a software product. Ian Sommerville details the four phases for a software process as being:

1. Specification
2. Development
3. Validation
4. Evolution

Different software development processes will impact the time distribution and therefore cost distributions for the different phases.

A waterfall process will have a longer specification phase and longer design time upfront. The development phase however will be shorter. The testing phase will be longer.

An iterative development process has a smaller time for specification and design. This is due to this work being done during the iterative development phase. As such the development phase is reduced. The testing phase for iterative development is often shorter because the system has been tested during development (at the end of an iteration).

Software engineering methods

A software engineering method is a structured approach to software development that aims to produce high quality software.

The attributes of good software are:

- High maintainability – Software should be developed in such a way that it can evolve to meet new/changing requirements.
- Reliability – Software should be bug free (as much as possible), secure and should not cause harm
- Efficiency - Software should not waste users time or system resources
- Usability – Software should be intuitive to use by its intended audience.

Software engineers have a wider responsibility than simply applying their technical skills. Software engineers also have responsibility to the profession and society.