CSC 221 – Introduction to Software Engineering

# systems engineering

extract from Sommerville's chapter 2 slides

Alan Dix

www.hcibook.com/alan/teaching/CSC221/



## Systems Engineering

- Designing, implementing, deploying and operating systems which include hardware, software and people



## What is a system?

- A purposeful collection of inter-related components working together towards some common objective.
- A system may include software, mechanical, electrical and electronic hardware and be operated by people.
- System components are dependent on other system components
- The properties and behaviour of system components are inextricably inter-mingled



## Problems of systems engineering

- Large systems are usually designed to solve 'wicked' problems
- Systems engineering requires a great deal of co-ordination across disciplines
  - Almost infinite possibilities for design trade-offs across components
  - Mutual distrust and lack of understanding across engineering disciplines
- Systems must be designed to last many years in a changing environment



## Software and systems engineering

- The proportion of software in systems is increasing. Software-driven general purpose electronics is replacing special-purpose systems
- Problems of systems engineering are similar to problems of software engineering
- Software is (unfortunately) seen as a problem in systems engineering. Many large system projects have been delayed because of software problems



## Emergent properties

- Properties of the system as a whole rather than properties that can be derived from the properties of components of a system
- Emergent properties are a consequence of the relationships between system components
- They can therefore only be assessed and measured once the components have been integrated into a system

# Examples of emergent properties

- *The overall weight of the system*
  - This is an example of an emergent property that can be computed from individual component properties.
- *The reliability of the system*
  - This depends on the reliability of system components and the relationships between the components.
- *The usability of a system*
  - This is a complex property which is not simply dependent on the system hardware and software but also depends on the system operators and the environment where it is used.



# Types of (emergent) property

- Functional properties
  - These appear when all the parts of a system work together to achieve some objective. For example, a bicycle has the functional property of being a transportation device once it has been assembled from its components.
- Non-functional (emergent) properties
  - Examples are reliability, performance, safety, and security. These relate to the behaviour of the system in its operational environment. They are often critical for computer-based systems as failure to achieve some minimal defined level in these properties may make the system unusable.



# The 'shall-not' properties

- Properties such as performance and reliability can be measured
- However, some properties are properties that the system should not exhibit
  - Safety - the system should not behave in an unsafe way
  - Security - the system should not permit unauthorised use
- Measuring or assessing these properties is very hard

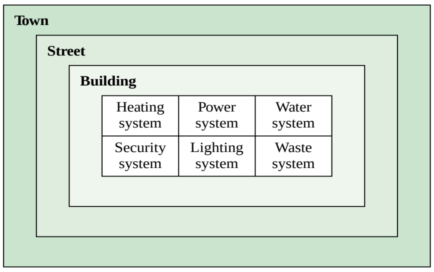


# Systems and their environment

- Systems are not independent but exist in an environment
- System's function may be to change its environment
- Environment affects the functioning of the system e.g. system may require electrical supply from its environment
- The organizational as well as the physical environment may be important



# System hierarchies






# Human and organisational factors

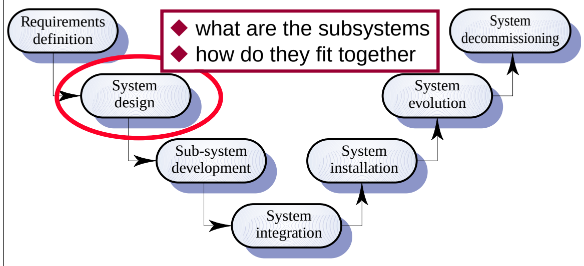
- *Process changes*
  - Does the system require changes to the work processes in the environment?
- *Job changes*
  - Does the system de-skill the users in an environment or cause them to change the way they work?
- *Organisational changes*
  - Does the system change the political power structure in an organisation?

# The system engineering process

- Usually follows a ‘waterfall’ model because of the need for parallel development of different parts of the system
  - Little scope for iteration between phases because hardware changes are very expensive. Software may have to compensate for hardware problems
- Inevitably involves engineers from different disciplines who must work together
  - Much scope for misunderstanding here. Different disciplines use a different vocabulary and much negotiation is required. Engineers may have personal agendas to fulfil



# The system engineering process






# The system design process

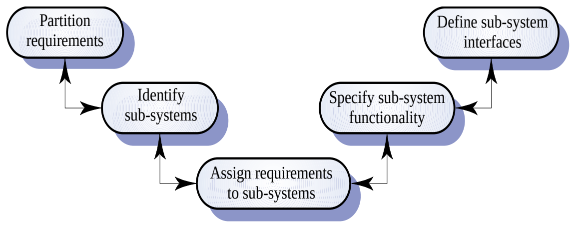
- Partition requirements
  - Organise requirements into related groups
- Identify sub-systems
  - Identify a set of sub-systems which collectively can meet the system requirements

N.B. emergent properties

- Assign requirements to sub-systems
  - Causes particular problems when COTS are integrated
- Specify sub-system functionality
- Define sub-system interfaces
  - Critical activity for parallel sub-system development



# The system design process






# System design problems

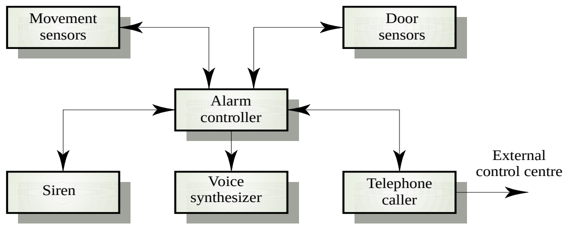
- Requirements partitioning to hardware, software and human components may involve a lot of negotiation
- Difficult design problems are often assumed to be readily solved using software
- Hardware platforms may be inappropriate for software requirements so software must compensate for this



# System architecture modelling

- An architectural model presents an abstract view of the sub-systems making up a system
- May include major information flows between sub-systems
- Usually presented as a block diagram
- May identify different types of functional component in the model

## Intruder alarm system






## Component types in alarm system

- Sensor
  - Movement sensor, door sensor
- Actuator
  - Siren
- Communication
  - Telephone caller
- Co-ordination
  - Alarm controller
- Interface
  - Voice synthesizer



## Functional system components

- Sensor components
- Actuator components
- Computation components **+ memory**
- Communication components
- Co-ordination components
- Interface components



## System components

- Sensor components
  - Collect information from the system's environment e.g. radars in an air traffic control system
- Actuator components
  - Cause some change in the system's environment e.g. valves in a process control system which increase or decrease material flow in a pipe
- Computation components
  - Carry out some computations on an input to produce an output e.g. a floating point processor in a computer system



## System components

- Communication components
  - Allow system components to communicate with each other e.g. network linking distributed computers
- Co-ordination components
  - Co-ordinate the interactions of other system components e.g. scheduler in a real-time system
- Interface components
  - Facilitate the interactions of other system components e.g. operator interface
- All components are now usually software controlled



## summary

- software is part of a larger system
  - hardware, software, people, environment
- emergent properties
  - the whole more than the sum of the parts
- system development
  - design subsystems and interrelations
  - hardware, software, people, environment