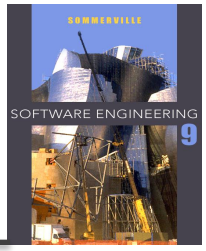
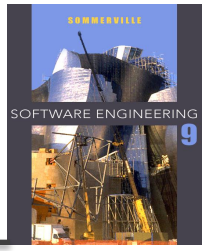


Chapter 10 – Sociotechnical Systems

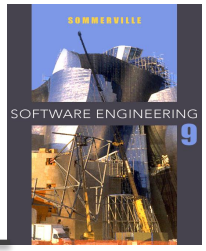
Lecture 1

Topics covered



- Complex systems
- Systems engineering
- Systems procurement
- System development
- System operation

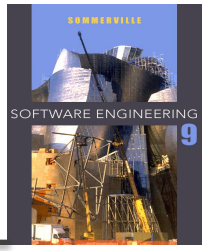
Systems



- Software engineering is not an isolated activity but is part of a broader systems engineering process.
- Software systems are therefore not isolated systems but are essential components of broader systems that have a human, social or organizational purpose.
- Example
  - Wilderness weather system is part of broader weather recording and forecasting systems
  - These include hardware and software, forecasting processes, system users, the organizations that depend on weather forecasts, etc.

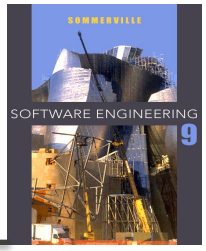
The sociotechnical systems stack

Layers in the STS stack



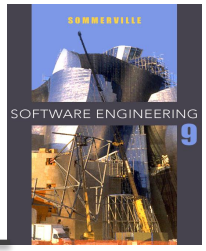
- Equipment
  - Hardware devices, some of which may be computers. Most devices will include an embedded system of some kind.
- Operating system
  - Provides a set of common facilities for higher levels in the system.
- Communications and data management
  - Middleware that provides access to remote systems and databases.
- Application systems
  - Specific functionality to meet some organization requirements.

Layers in the STS stack



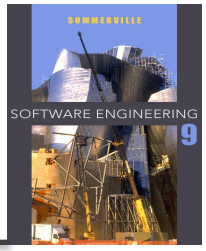
- Business processes
  - A set of processes involving people and computer systems that support the activities of the business.
- Organizations
  - Higher level strategic business activities that affect the operation of the system.
- Society
  - Laws, regulation and culture that affect the operation of the system.

Holistic system design



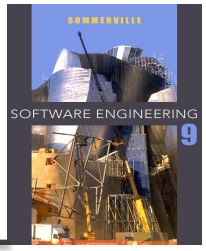
- There are interactions and dependencies between the layers in a system and changes at one level ripple through the other levels
  - Example: Change in regulations (society) leads to changes in business processes and application software.
- For dependability, a systems perspective is essential
  - Contain software failures within the enclosing layers of the STS stack.
  - Understand how faults and failures in adjacent layers may affect the software in a system.

Complex systems



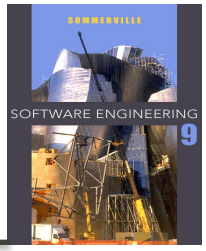
- A system is a purposeful collection of inter-related components working together to achieve some common objective.
- A system may include software, mechanical, electrical and electronic hardware and be operated by people.
- System components are dependent on other system components.
- The properties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

System categories



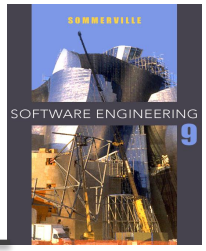
- Technical computer-based systems
  - Systems that include hardware and software but where the operators and operational processes are not normally considered to be part of the system. The system is not self-aware.
  - Example: A word processor used to write a book.
- Socio-technical systems
  - Systems that include technical systems but also operational processes and people who use and interact with the technical system. Socio-technical systems are governed by organisational policies and rules.
  - Example: A publishing system to produce a book.

Organizational affects



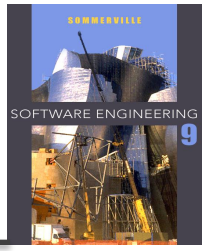
- Process changes
  - Systems may require changes to business processes so training may be required. Significant changes may be resisted by users.
- Job changes
  - Systems may de-skill users or cause changes to the way they work. The status of individuals in an organization may be affected by the introduction of a new system.
- Organizational changes
  - Systems may change the political power structure in an organization. If an organization depends on a system then those that control the system have more power.

Socio-technical system characteristics



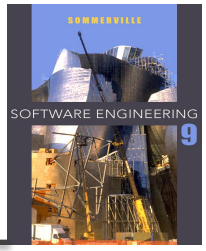
- Emergent properties
  - Properties of the system of a whole that depend on the system components and their relationships.
- Non-deterministic
  - They do not always produce the same output when presented with the same input because the systems's behaviour is partially dependent on human operators.
- Complex relationships with organisational objectives
  - The extent to which the system supports organisational objectives does not just depend on the system itself.

Emergent properties



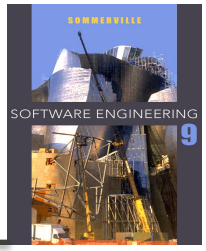
- Properties of the system as a whole rather than properties that can be derived from the properties of components of a system
- Emergent properties are a consequence of the relationships between system components
- They can therefore only be assessed and measured once the components have been integrated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

Examples of emergent properties



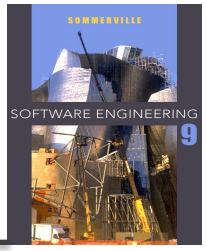
Property	Description
Volume	The volume of a system (the total space occupied) varies depending on how the component assemblies are arranged and connected.
Reliability	System reliability depends on component reliability but unexpected interactions can cause new types of failures and therefore affect the reliability of the system.
Security	The security of the system (its ability to resist attack) is a complex property that cannot be easily measured. Attacks may be devised that were not anticipated by the system designers and so may defeat built-in safeguards.
Repairability	This property reflects how easy it is to fix a problem with the system once it has been discovered. It depends on being able to diagnose the problem, access the components that are faulty, and modify or replace these components.
Usability	This property reflects how easy it is to use the system. It depends on the technical system components, its operators, and its operating environment.

Types of emergent property



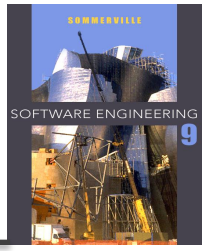
- Functional properties
  - These appear when all the parts of a system work together to achieve some objective. For example, a bicycle has the functional property of being a transportation device once it has been assembled from its components.
- Non-functional emergent properties
  - Examples are reliability, 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.

Reliability as an emergent property



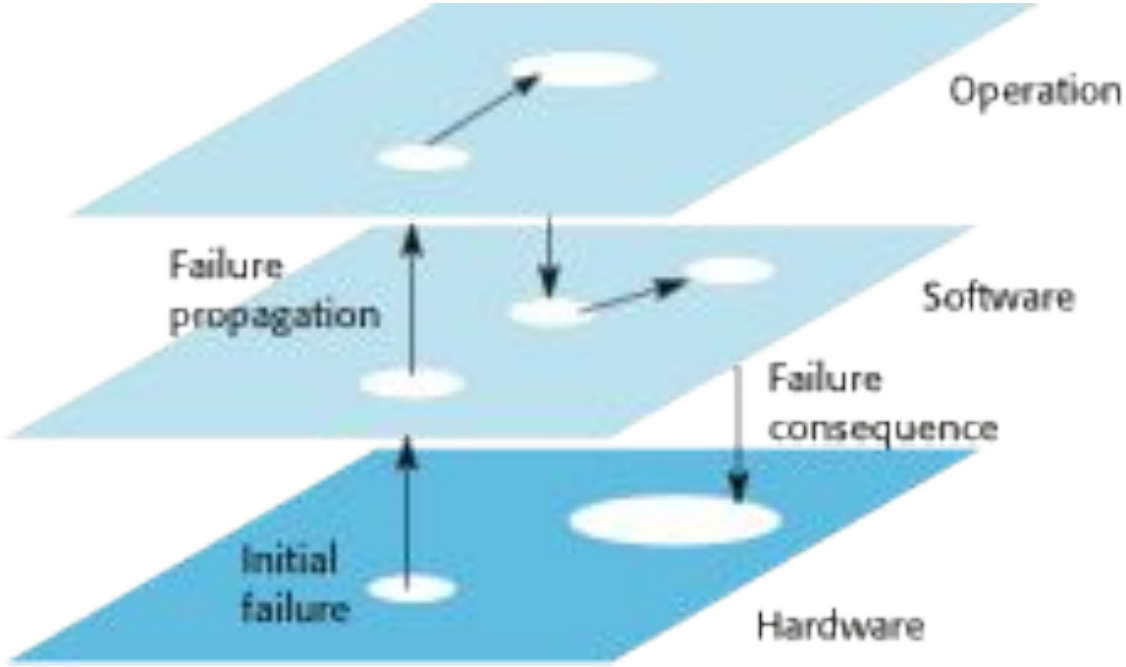
- Because of component inter-dependencies, faults can be propagated through the system.
- System failures often occur because of unforeseen inter-relationships between components.
- It is practically impossible to anticipate all possible component relationships.
- Software reliability measures may give a false picture of the overall system reliability.

Influences on reliability

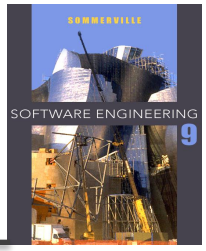


- *Hardware reliability*
  - What is the probability of a hardware component failing and how long does it take to repair that component?
- *Software reliability*
  - How likely is it that a software component will produce an incorrect output. Software failure is usually distinct from hardware failure in that software does not wear out.
- *Operator reliability*
  - How likely is it that the operator of a system will make an error?
- Failures are not independent and they propagate from one level to another.

Failure propagation

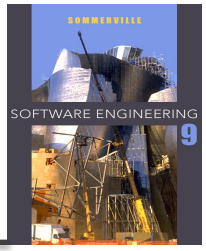


Non-determinism



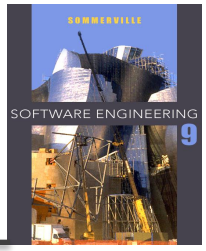
- A deterministic system is one where a given sequence of inputs will always produce the same sequence of outputs.
- Software systems are deterministic; systems that include humans are non-deterministic
  - A socio-technical system will not always produce the same sequence of outputs from the same input sequence
  - Human elements
    - People do not always behave in the same way
  - System changes
    - System behaviour is unpredictable because of frequent changes to hardware, software and data.

Success criteria



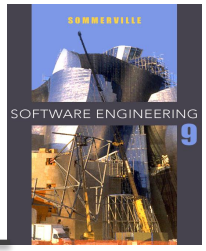
- Complex systems are developed to address 'wicked problems' – problems where there cannot be a complete specification.
- Different stakeholders see the problem in different ways and each has a partial understanding of the issues affecting the system.
- Consequently, different stakeholders have their own views about whether or not a system is 'successful'
  - Success is a judgment and cannot be objectively measured.
  - Success is judged using the effectiveness of the system when deployed rather than judged against the original reasons for procuement.

Conflicting views of success



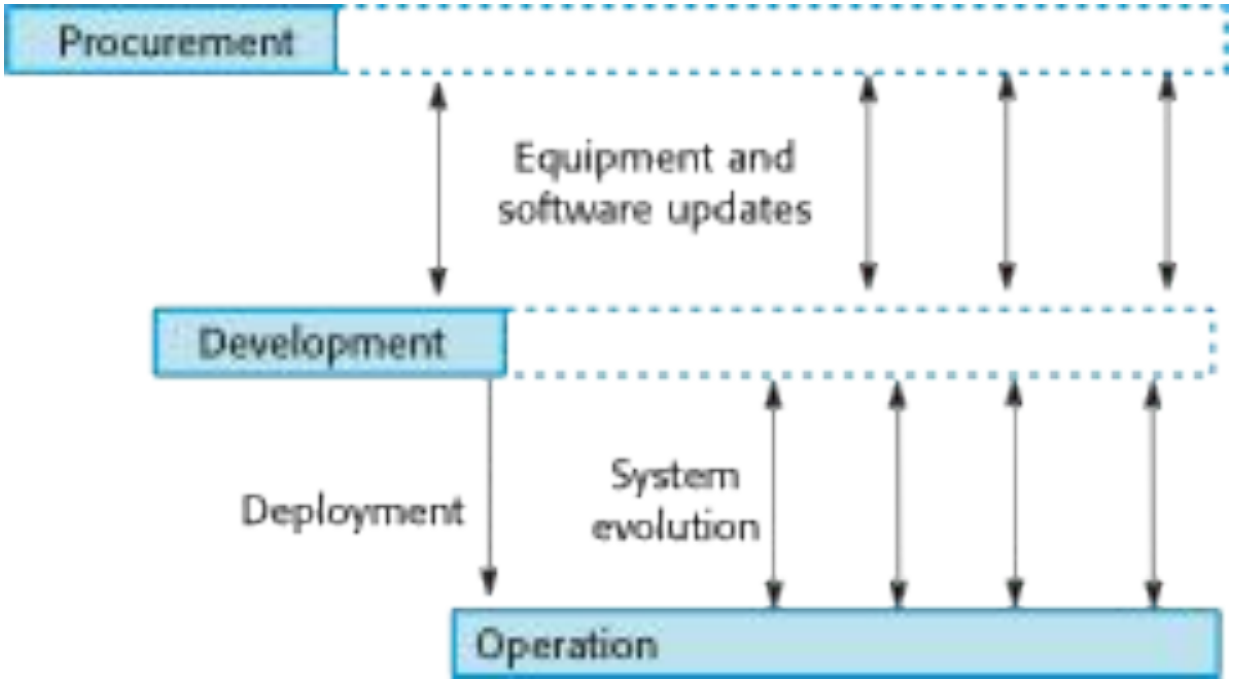
- MHC-PMS designed to support multiple, conflicting goals
  - Improve quality of care.
  - Provide better information and care costs and so increase revenue.
- Fundamental conflict
  - To satisfy reporting goal, doctors and nurses had to provide additional information over and above that required for clinical purposes.
  - They had less time to interact with patients, so quality of care reduced. System was not a success.
- However, managers had better reports
  - System was a success from a managerial perspective.

Systems engineering

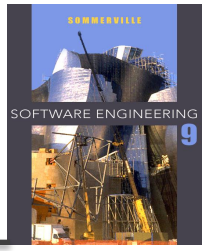


- Procuring, specifying, designing, implementing, validating, deploying and maintaining socio-technical systems.
- Concerned with the services provided by the system, constraints on its construction and operation and the ways in which it is used to fulfil its purpose or purposes.

Stages of systems engineering

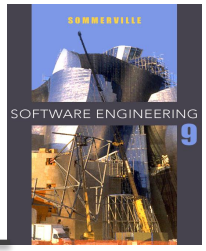


Systems engineering stages



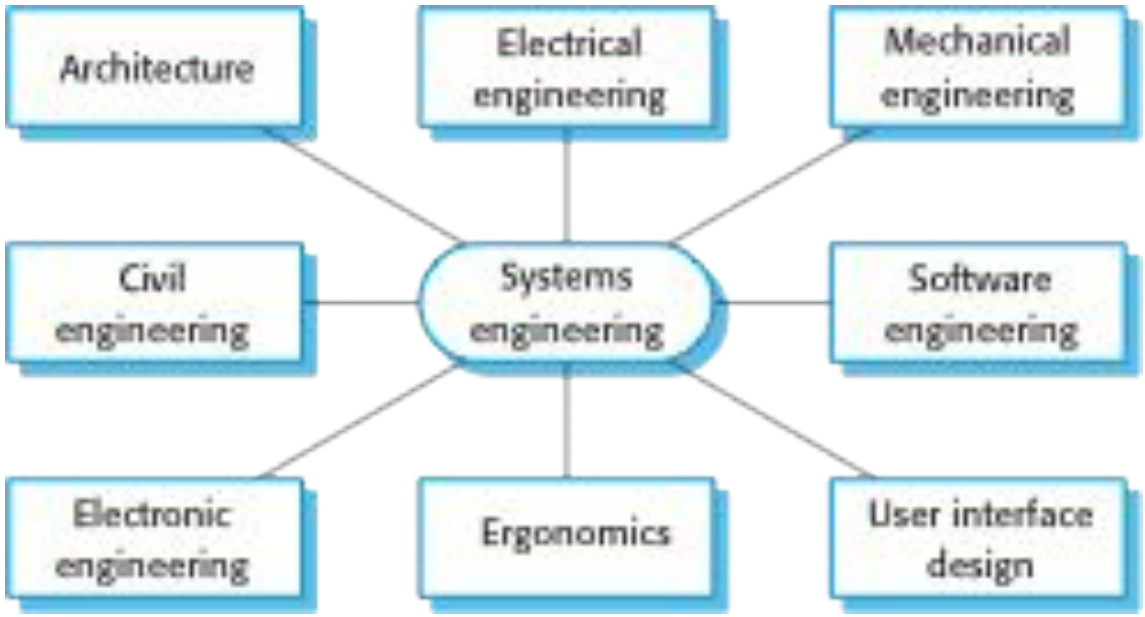
- Procurement (acquisition)
  - The purpose of the system is established, high-level system requirements are defined, decisions are made on how functionality is distributed and the system components are purchased.
- Development
  - The system is developed – requirements are defined in detail, the system is implemented and tested and operational processes are defined.
- Operation
  - The system is deployed and put into use. Changes are made as new requirements emerge. Eventually, the system is decommissioned.

Security and dependability considerations

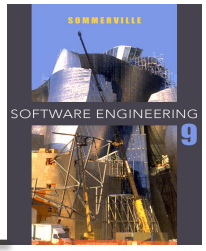


- Design options limited by procurement decisions
  - Purchased components may make some safeguards impossible to implement.
- Human errors made during development may introduce faults into the system.
- Inadequate testing may mean faults are not discovered before deployment.
- Configuration errors during deployment may introduce vulnerabilities.
- Assumptions made during procurement may be forgotten when system changes are made.

Professional disciplines involved in systems engineering

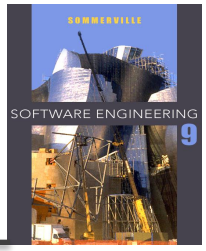


Inter-disciplinary working

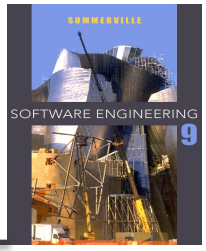


- Communication difficulties
  - Different disciplines use the same terminology to mean different things. This can lead to misunderstandings about what will be implemented.
- Differing assumptions
  - Each discipline makes assumptions about what can and can't be done by other disciplines.
- Professional boundaries
  - Each discipline tries to protect their professional boundaries and expertise and this affects their judgments on the system.

Key points



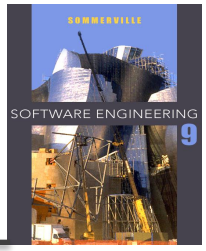
- Socio-technical systems include computer hardware, software and people and are designed to meet some business goal.
- Human and organizational factors, such as the organizational structure, have a significant effect on the operation of socio-technical systems.
- Emergent properties are properties that are characteristic of the system as a whole and not its component parts.
- The fundamental stages of systems engineering are procurement, development and operation.



Chapter 10 – Sociotechnical Systems

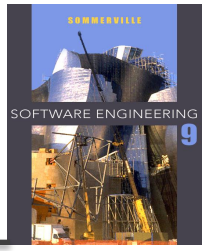
Lecture 2

System procurement



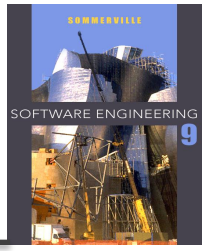
- Acquiring a system (or systems) to meet some identified organizational need.
- Before procurement, decisions are made on:
  - Scope of the system
  - System budgets and timescales
  - High-level system requirements
- Based on this information, decisions are made on whether to procure a system, the type of system and the potential system suppliers.

Decision drivers



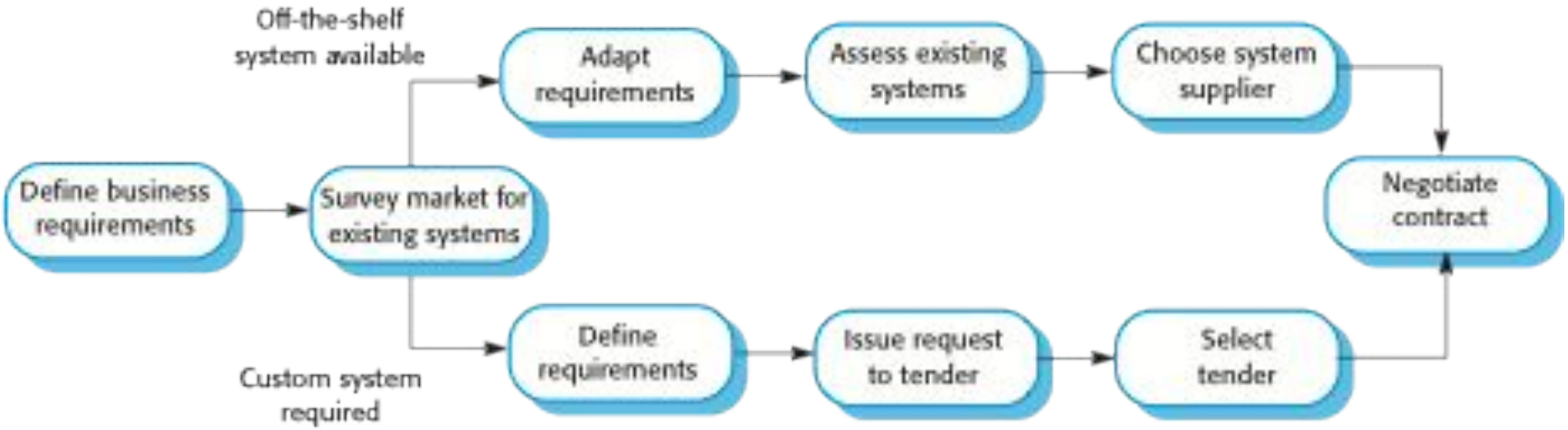
- The state of other organizational systems
- The need to comply with external regulations
- External competition
- Business re-organization
- Available budget

Procurement and development

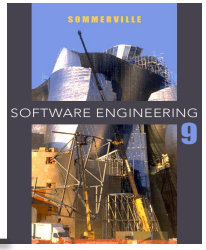


- Some system specification and architectural design is usually necessary before procurement
  - You need a specification to let a contract for system development
  - The specification may allow you to buy a commercial off-the-shelf (COTS) system. Almost always cheaper than developing a system from scratch
- Large complex systems usually consist of a mix of off the shelf and specially designed components. The procurement processes for these different types of component are usually different.

System procurement processes

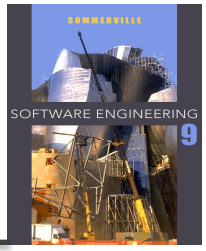


Procurement issues



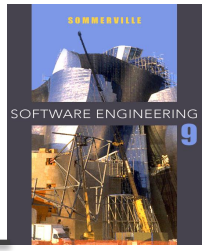
- Requirements may have to be modified to match the capabilities of off-the-shelf components.
- The requirements specification may be part of the contract for the development of the system.
- There is usually a contract negotiation period to agree changes after the contractor to build a system has been selected.

Contractors and sub-contractors



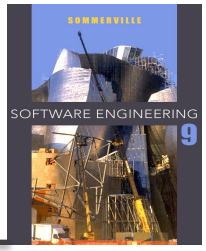
- The procurement of large hardware/software systems is usually based around some principal contractor.
- Sub-contracts are issued to other suppliers to supply parts of the system.
- Customer liases with the principal contractor and does not deal directly with sub-contractors.

Procurement and dependability



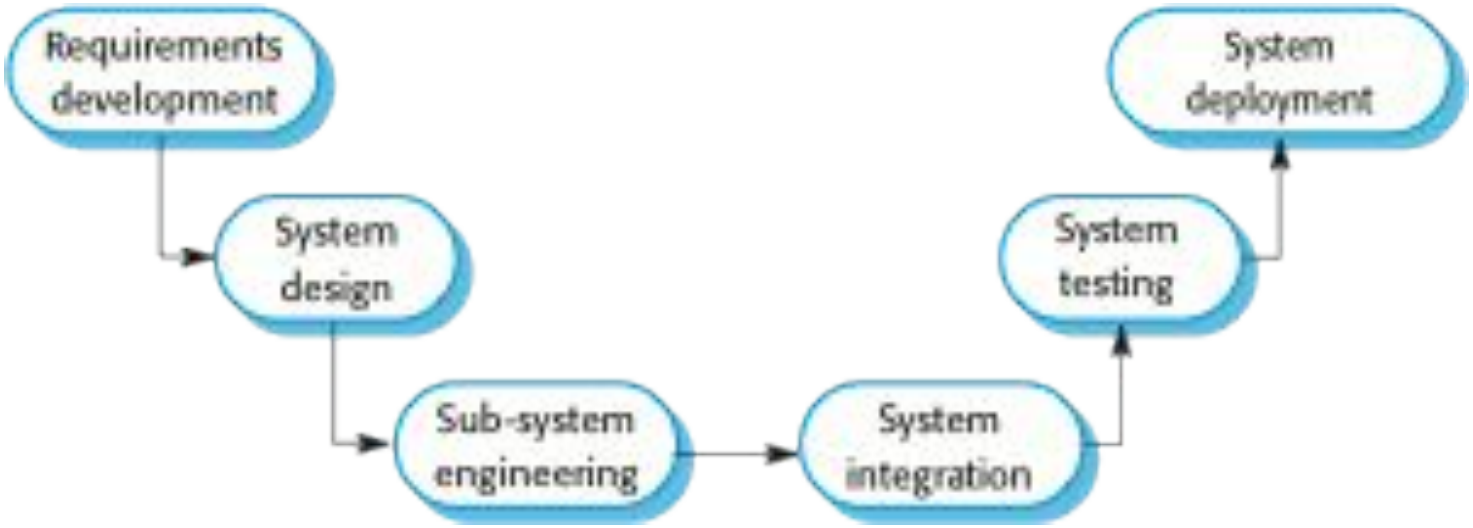
- Procurement decisions have profound effects on system dependability as these decisions limit the scope of dependability requirements.
- For an off-the-shelf system, the procurer has very limited influence on the security and dependability requirements of the system.
- For a custom system, considerable effort has to be expended in defining security and dependability requirements.

System development

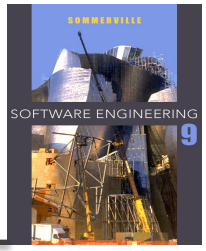


- Usually follows a plan-driven approach because of the need for parallel development of different parts of the system
  - Little scope for iteration between phases because hardware changes are very expensive. Software may have to compensate for hardware problems.
- Inevitably involves engineers from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

Systems development

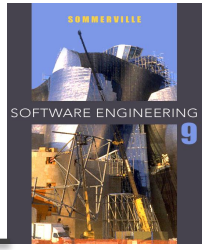


System requirements definition



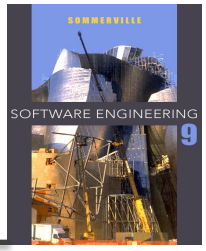
- Three types of requirement defined at this stage
  - Abstract functional requirements. System functions are defined in an abstract way;
  - System properties. Non-functional requirements for the system in general are defined;
  - Undesirable characteristics. Unacceptable system behaviour is specified.
- Should also define overall organisational objectives for the system.

The system design process



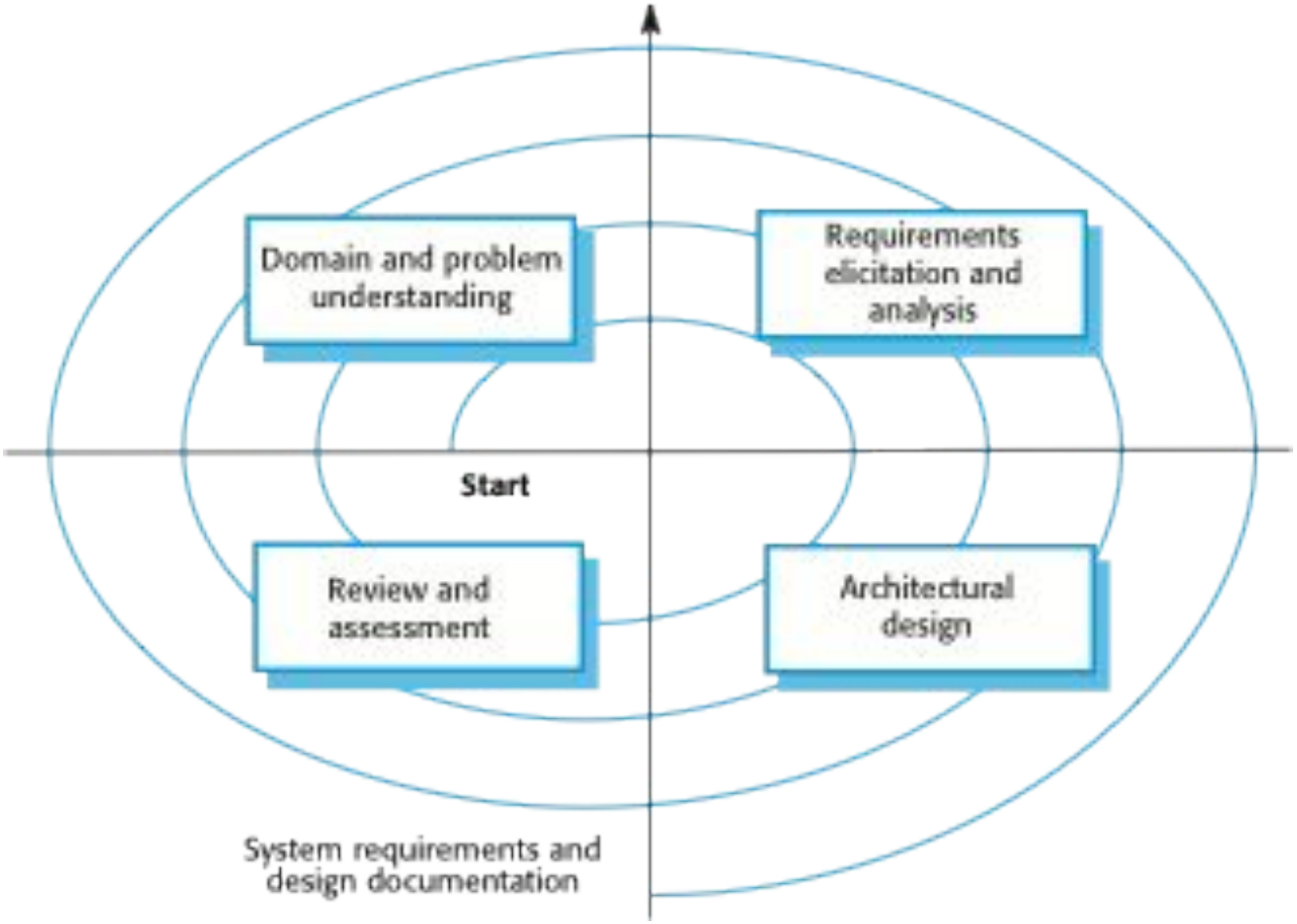
- Partition requirements
  - Organise requirements into related groups.
- Identify sub-systems
  - Identify a set of sub-systems which collectively can meet the system requirements.
- Assign requirements to sub-systems
  - Causes particular problems when COTS are integrated.
- Specify sub-system functionality.
- Define sub-system interfaces
  - Critical activity for parallel sub-system development.

Requirements and design

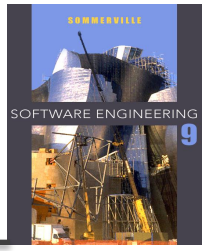


- Requirements engineering and system design are inextricably linked.
- Constraints posed by the system's environment and other systems limit design choices so the actual design to be used may be a requirement.
- Initial design may be necessary to structure the requirements.
- As you do design, you learn more about the requirements.

Requirements and design spiral

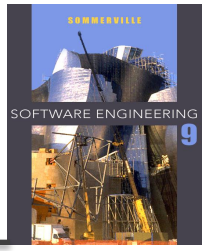


Sub-system development



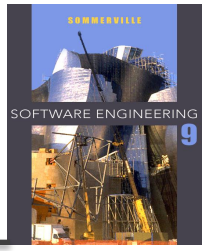
- Typically parallel projects developing the hardware, software and communications.
- May involve some COTS (Commercial Off-the-Shelf) systems procurement.
- Lack of communication across implementation teams can cause problems.
- There may be a bureaucratic and slow mechanism for proposing system changes, which means that the development schedule may be extended because of the need for rework.

System integration



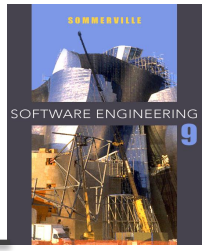
- The process of putting hardware, software and people together to make a system.
- Should ideally be tackled incrementally so that sub-systems are integrated one at a time.
- The system is tested as it is integrated.
- Interface problems between sub-systems are usually found at this stage.
- May be problems with uncoordinated deliveries of system components.

System delivery and deployment



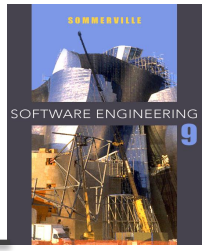
- After completion, the system has to be installed in the customer's environment
  - Environmental assumptions may be incorrect;
  - May be human resistance to the introduction of a new system;
  - System may have to coexist with alternative systems for some time;
  - May be physical installation problems (e.g. cabling problems);
  - Data cleanup may be required;
  - Operator training has to be identified.

Development and dependability



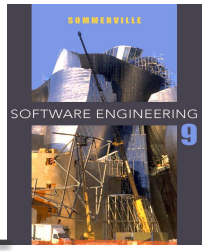
- Decisions are made on dependability and security requirements and trade-offs made between costs, schedule, performance and dependability.
- Human errors may lead to the introduction of faults into the system.
- Testing and validation processes may be limited because of limited budgets.
- Problems in deployment mean there may be a mismatch between the system and its operational environment.

System operation



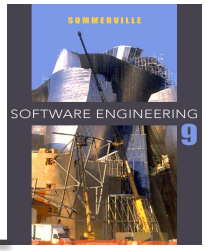
- Operational processes are the processes involved in using the system for its defined purpose.
- For new systems, these processes may have to be designed and tested and operators trained in the use of the system.
- Operational processes should be flexible to allow operators to cope with problems and periods of fluctuating workload.

Human error



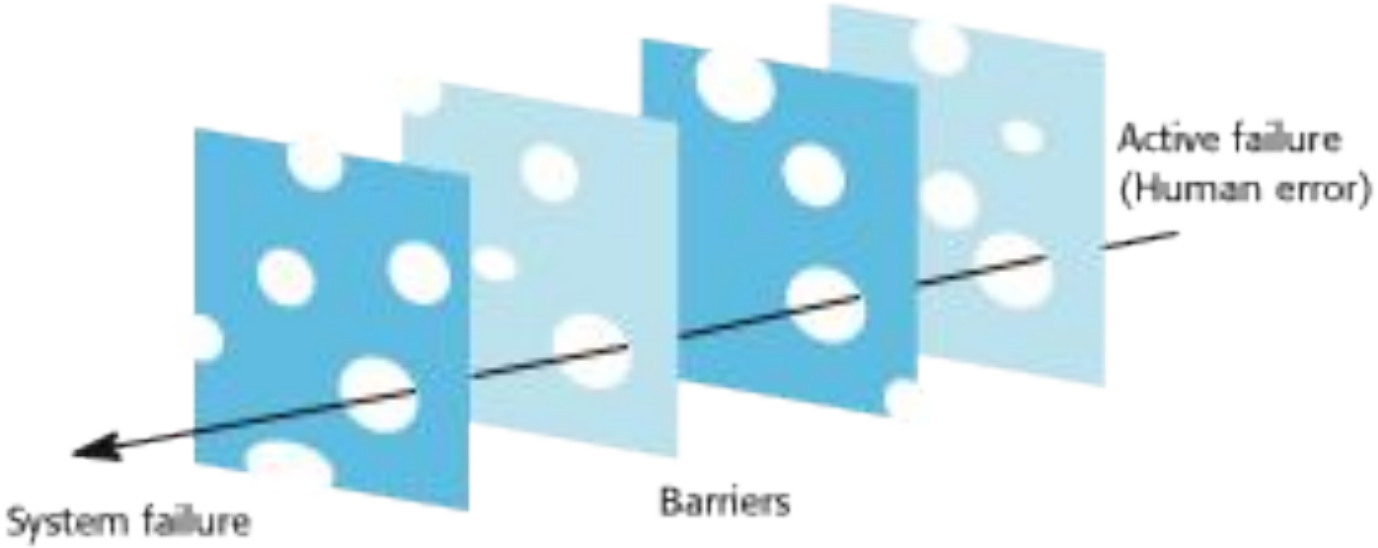
- Human errors occur in operational processes that influence the overall dependability of the system.
- Viewing human errors:
  - The person approach makes errors the responsibility of the individual and places the blame for error on the operator concerned. Actions to reduce error include threats of punishment, better training, more stringent procedures, etc.
  - The systems approach assumes that people are fallible and will make mistakes. The system is designed to detect these mistakes before they lead to system failure. When a failure occurs, the aim is not to blame an individual but to understand why the system defenses did not trap the error.

System defenses

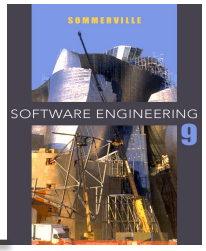


- To improve security and dependability, designers should think about the checks for human error that should be included in a system.
- As I discuss in later lectures, there should be multiple (redundant) barriers which should be different (diverse)
- No single barrier can be perfect.
  - There will be latent conditions in the system that may lead to failure.
- However, with multiple barriers, all have to fail for a system failure to occur.

Reason's Swiss cheese model of system failure

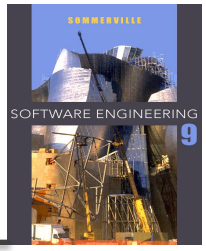


Defenses in an ATC system



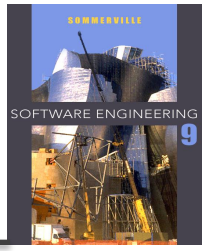
- Conflict alert system
  - Raises an audible alarm when aircraft are on conflicting paths
- Recording of instructions
  - Allows instructions issues to be reviewed and checked.
- Sharing of information
  - The team of controllers cross-check each other's work.

System evolution



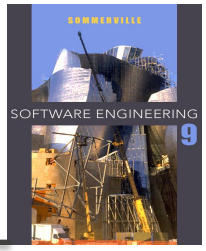
- Large systems have a long lifetime. They must evolve to meet changing requirements.
- Evolution is inherently costly
  - Changes must be analysed from a technical and business perspective;
  - Sub-systems interact so unanticipated problems can arise;
  - There is rarely a rationale for original design decisions;
  - System structure is corrupted as changes are made to it.
- Existing systems which must be maintained are sometimes called legacy systems.

Evolution and dependability



- Changes to a system are often a source of problems and vulnerabilities.
- Changes may be made without knowledge of previous design decisions made for security and dependability reasons.
  - Built-in safeguards may stop working.
- New faults may be introduced or latent faults exposed by changes.
  - These may not be discovered because complete system retesting is too expensive.

Key points



- System procurement covers all of the activities involved in deciding what system to buy and who should supply that system.
- System development includes requirements specification, design, construction, integration and testing.
- When a system is put into use, the operational processes and the system itself have to change to reflect changing business requirements.
- Human errors are inevitable and systems should include barriers to detect these errors before they lead to system failure.