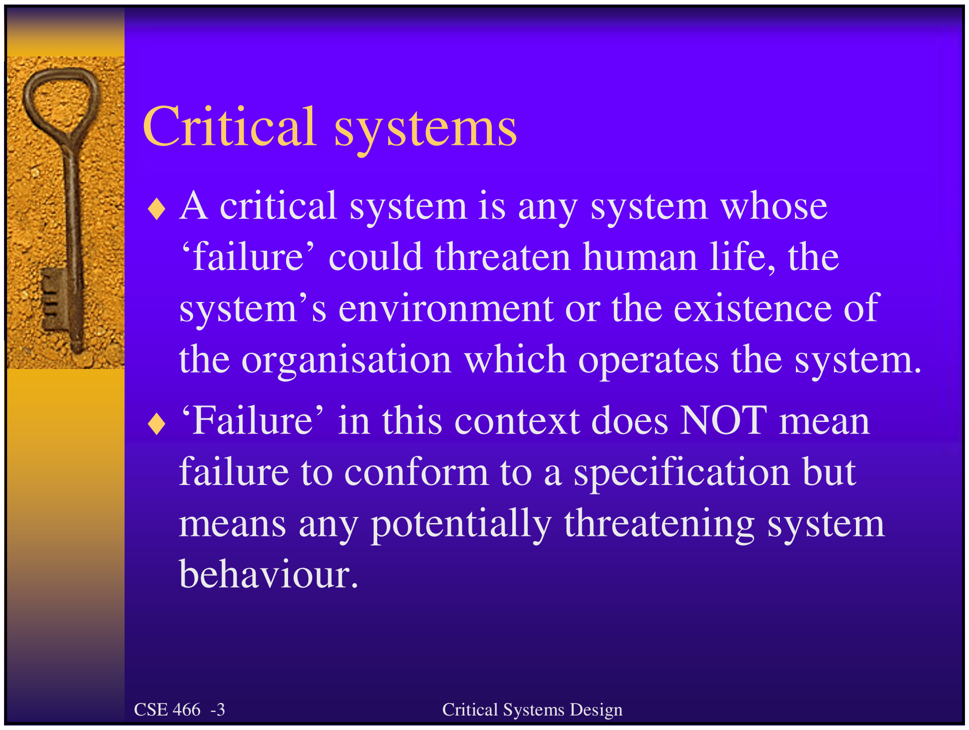
Critical Systems Engineering

- Engineering systems to avoid disasters

Adapted from Ian Sommerville

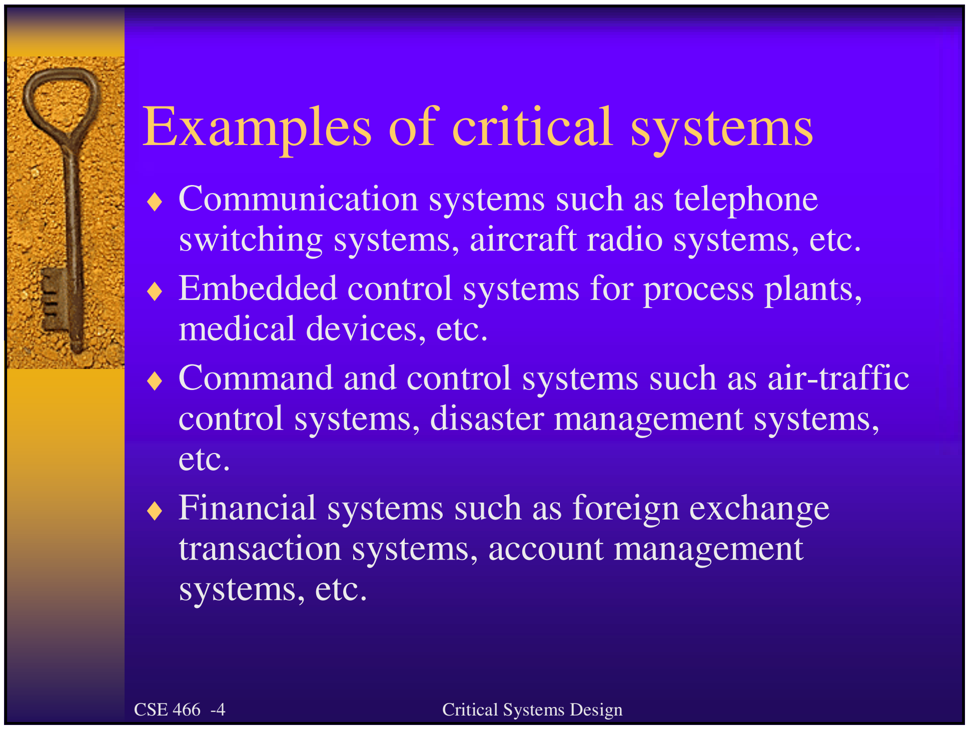
Objectives

- To introduce the notion of critical systems
- To describe critical system attributes (reliability, availability, maintainability, safety and security)
- To introduce techniques used for developing reliable and safe systems
- To discuss the importance of people in critical systems engineering



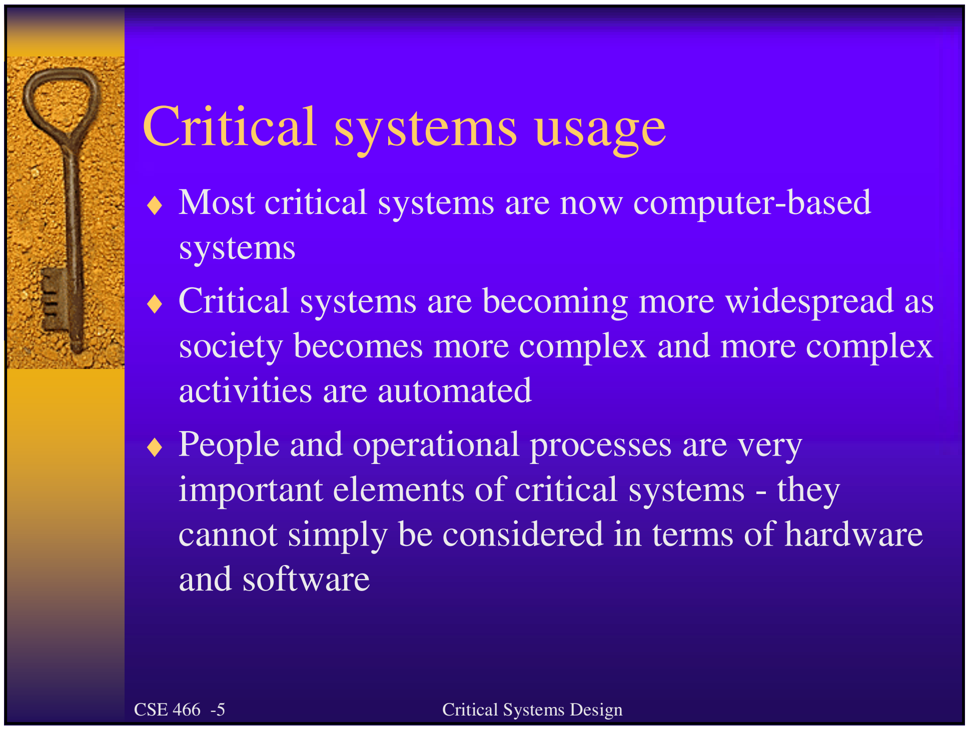
Critical systems

- A critical system is any system whose ‘failure’ could threaten human life, the system’s environment or the existence of the organisation which operates the system.
- ‘Failure’ in this context does NOT mean failure to conform to a specification but means any potentially threatening system behaviour.



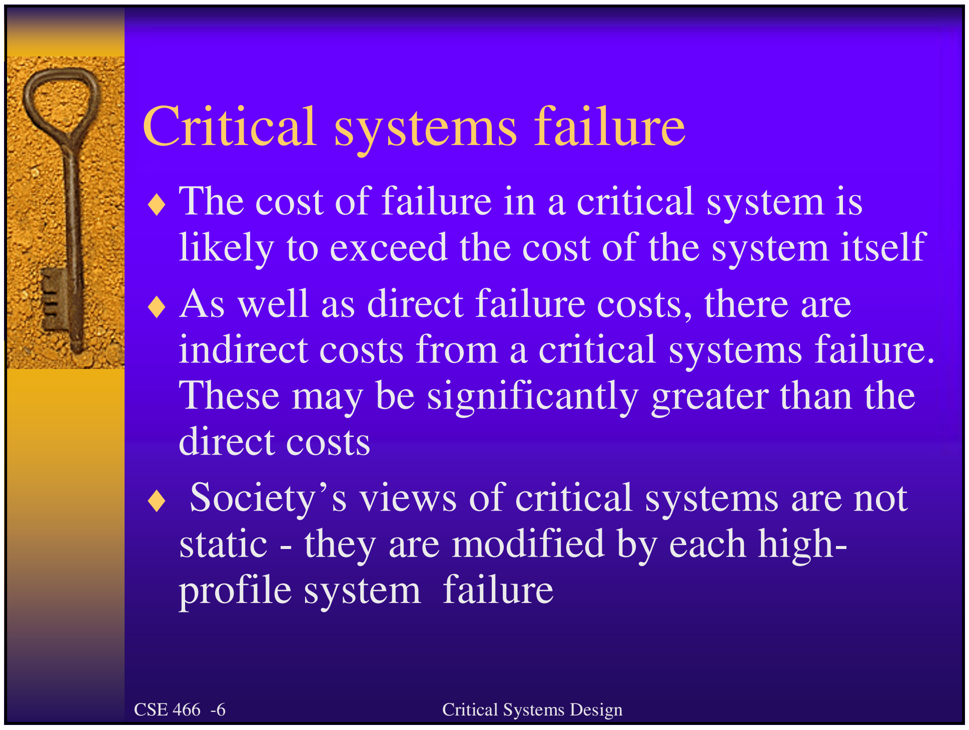
Examples of critical systems

- Communication systems such as telephone switching systems, aircraft radio systems, etc.
- Embedded control systems for process plants, medical devices, etc.
- Command and control systems such as air-traffic control systems, disaster management systems, etc.
- Financial systems such as foreign exchange transaction systems, account management systems, etc.



Critical systems usage

- Most critical systems are now computer-based systems
- Critical systems are becoming more widespread as society becomes more complex and more complex activities are automated
- People and operational processes are very important elements of critical systems - they cannot simply be considered in terms of hardware and software



Critical systems failure

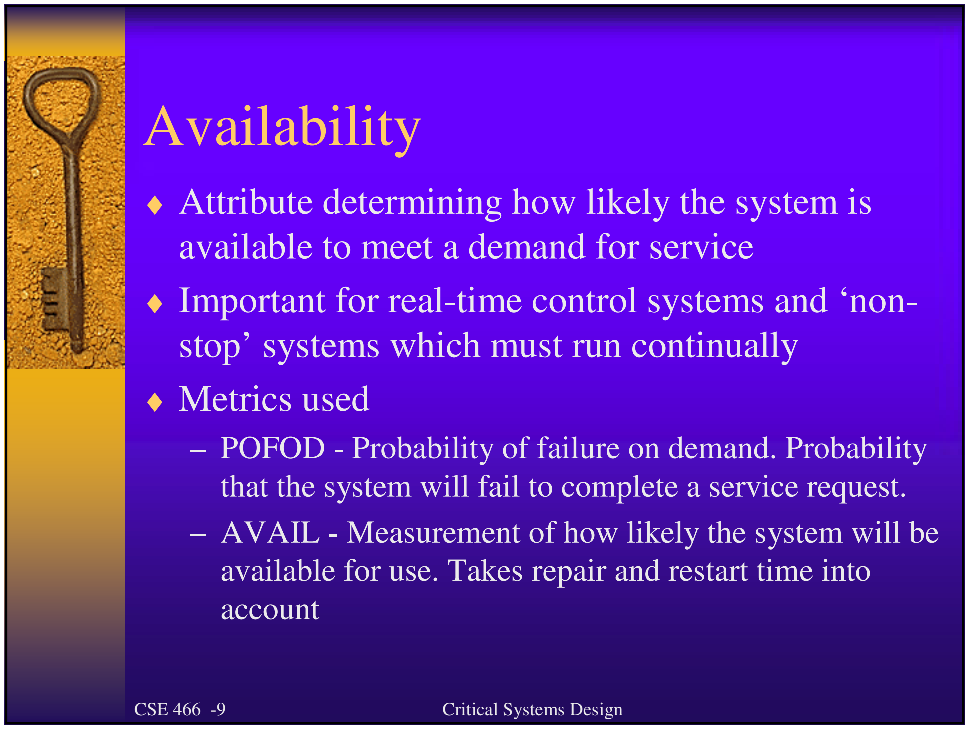
- The cost of failure in a critical system is likely to exceed the cost of the system itself
- As well as direct failure costs, there are indirect costs from a critical systems failure. These may be significantly greater than the direct costs
- Society’s views of critical systems are not static - they are modified by each high-profile system failure

Criticality attributes

- Reliability
  - Concerned with failure to perform to specification
- Availability
  - Concerned with failure to deliver required services
- Maintainability
  - Concerned with the ability of the system to evolve
- Safety
  - Concerned with behaviour which directly or indirectly threatens human life
- Security
  - Concerned with the ability of the system to protect itself

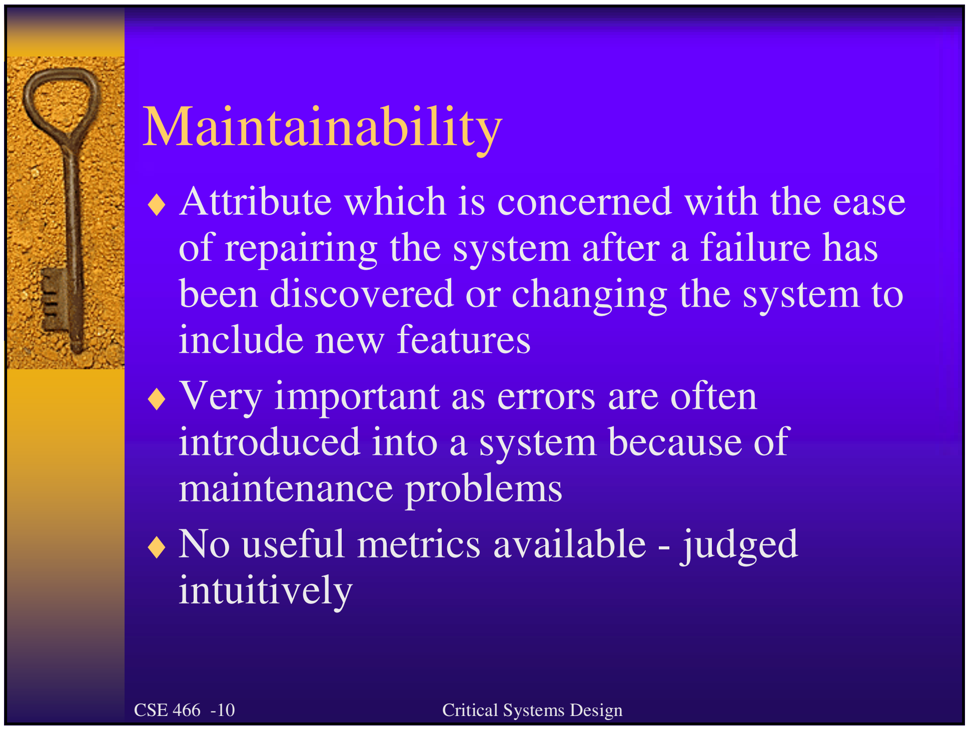
Reliability

- Attribute concerned with the number of times a system fails to deliver specified services. Difficult to define in an intuitive way
- Can’t be defined without defining the context of use of the system
- Metrics used
  - MTTF - Mean Time to Failure. Time between observed system failures
  - ROCOF - Rate of occurrence of failures. Number of failures in a given time period



Availability

- Attribute determining how likely the system is available to meet a demand for service
- Important for real-time control systems and ‘non-stop’ systems which must run continually
- Metrics used
  - POFOD - Probability of failure on demand. Probability that the system will fail to complete a service request.
  - AVAIL - Measurement of how likely the system will be available for use. Takes repair and restart time into account



Maintainability

- Attribute which is concerned with the ease of repairing the system after a failure has been discovered or changing the system to include new features
- Very important as errors are often introduced into a system because of maintenance problems
- No useful metrics available - judged intuitively

Safety

- Attribute concerned with the system’s ability to deliver its services in such a way the human life or the system’s environment will not be damaged by the system
- Increasingly important as computer-based systems take over functions which were previously performed by people
- Very difficult to specify and assess

Security

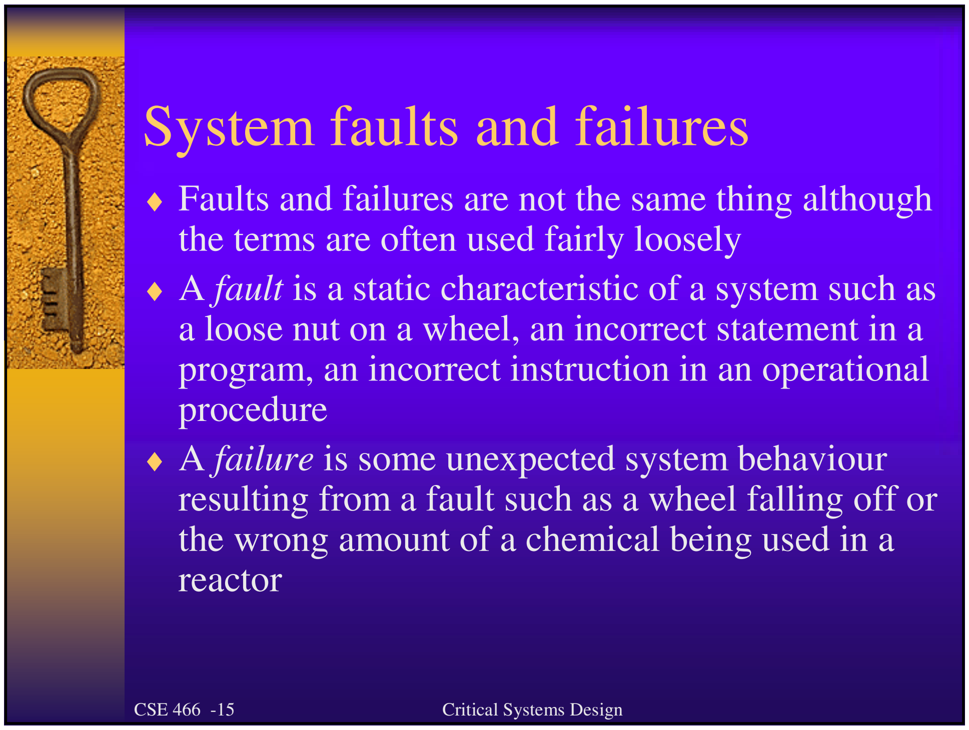
- Attribute concerned with the ability of the system to protect itself and its data from deliberate or accidental damage
- Affected by other attributes as a security failure can compromise reliability, availability and safety
- Similar in many respects to safety as security problems often arise because of events where were not anticipated by the specifiers/designers of the system

Critical systems development

- Critical systems attributes are NOT independent - the systems development process must be organised so that all of them are satisfied at least to some minimum level
- More rigorous (and expensive) development techniques have to be used for critical systems development because of the potential cost of failure

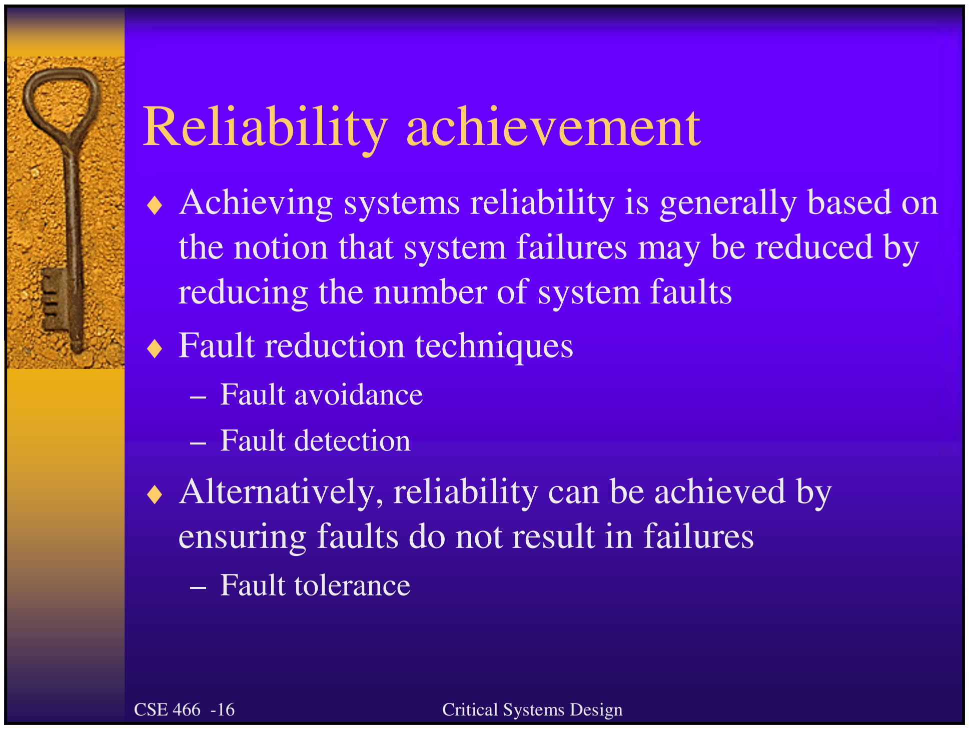
Developing reliable systems

- Reliable systems should be ‘fault-free’ systems where ‘fault-free’ means that the system’s behaviour always conforms to its specification
- Systems which are ‘fault-free’ may still fail because of specification or operational errors
- The costs of producing reliable systems grows exponentially as reliability requirements are increased. In reality, we can never be sure that we have produced a ‘fault-free’ system



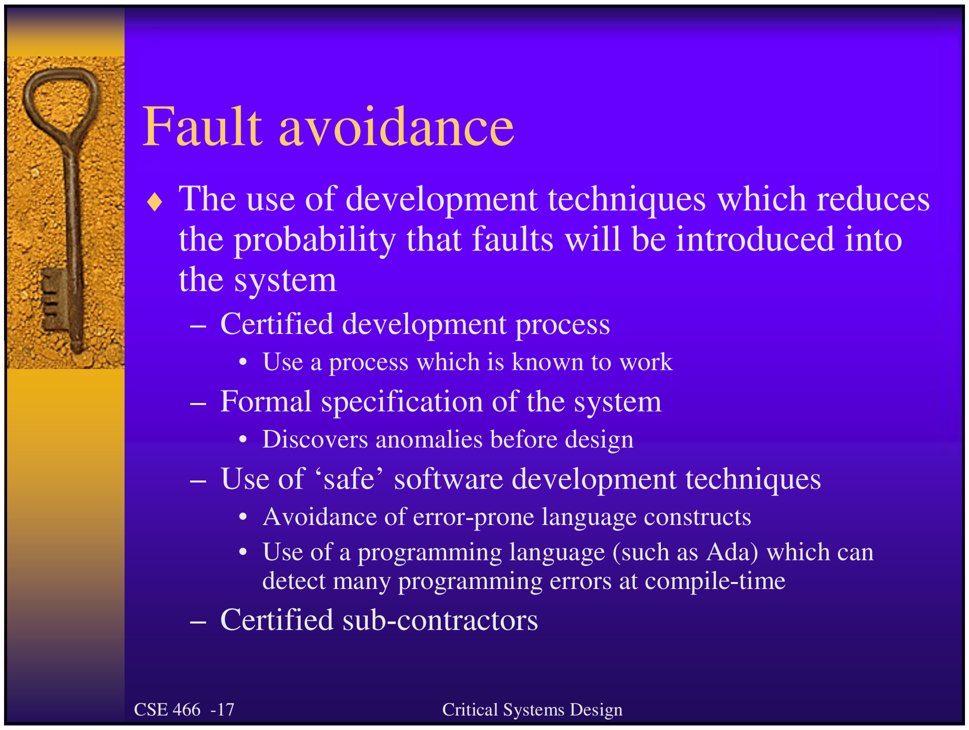
System faults and failures

- Faults and failures are not the same thing although the terms are often used fairly loosely
- A *fault* is a static characteristic of a system such as a loose nut on a wheel, an incorrect statement in a program, an incorrect instruction in an operational procedure
- A *failure* is some unexpected system behaviour resulting from a fault such as a wheel falling off or the wrong amount of a chemical being used in a reactor



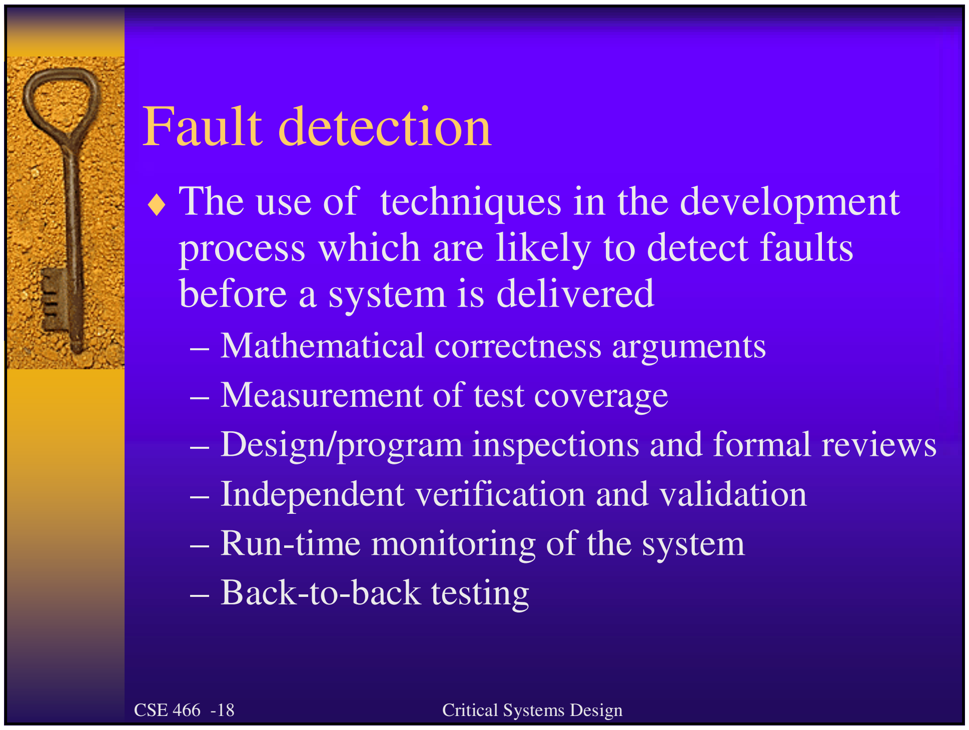
Reliability achievement

- Achieving systems reliability is generally based on the notion that system failures may be reduced by reducing the number of system faults
- Fault reduction techniques
  - Fault avoidance
  - Fault detection
- Alternatively, reliability can be achieved by ensuring faults do not result in failures
  - Fault tolerance



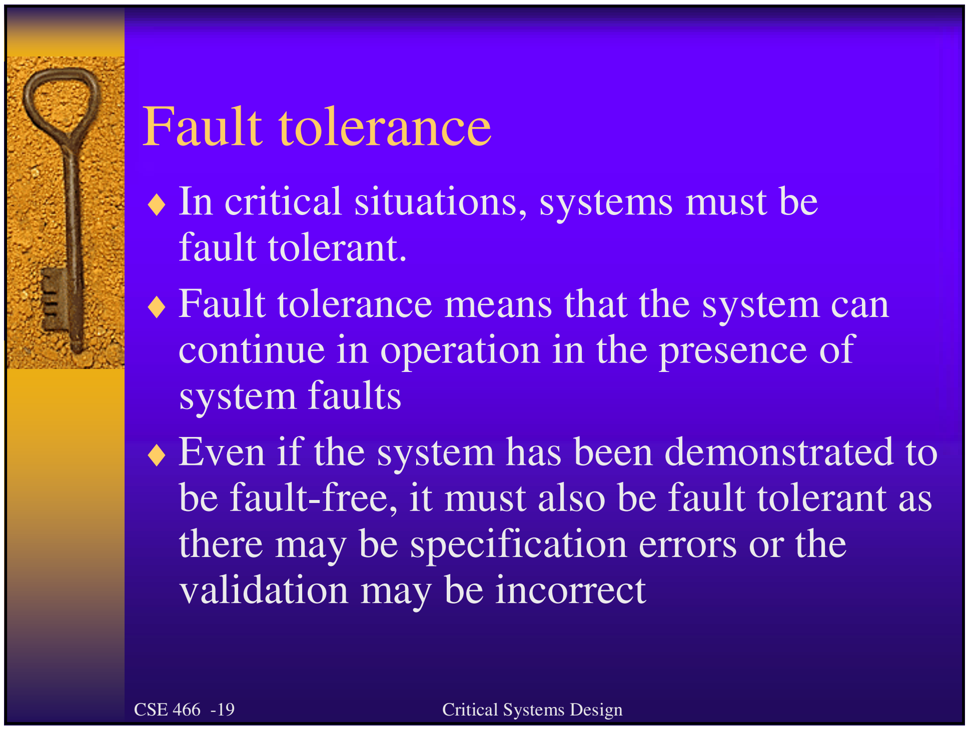
Fault avoidance

- The use of development techniques which reduces the probability that faults will be introduced into the system
  - Certified development process
    - Use a process which is known to work
  - Formal specification of the system
    - Discovers anomalies before design
  - Use of ‘safe’ software development techniques
    - Avoidance of error-prone language constructs
    - Use of a programming language (such as Ada) which can detect many programming errors at compile-time
  - Certified sub-contractors



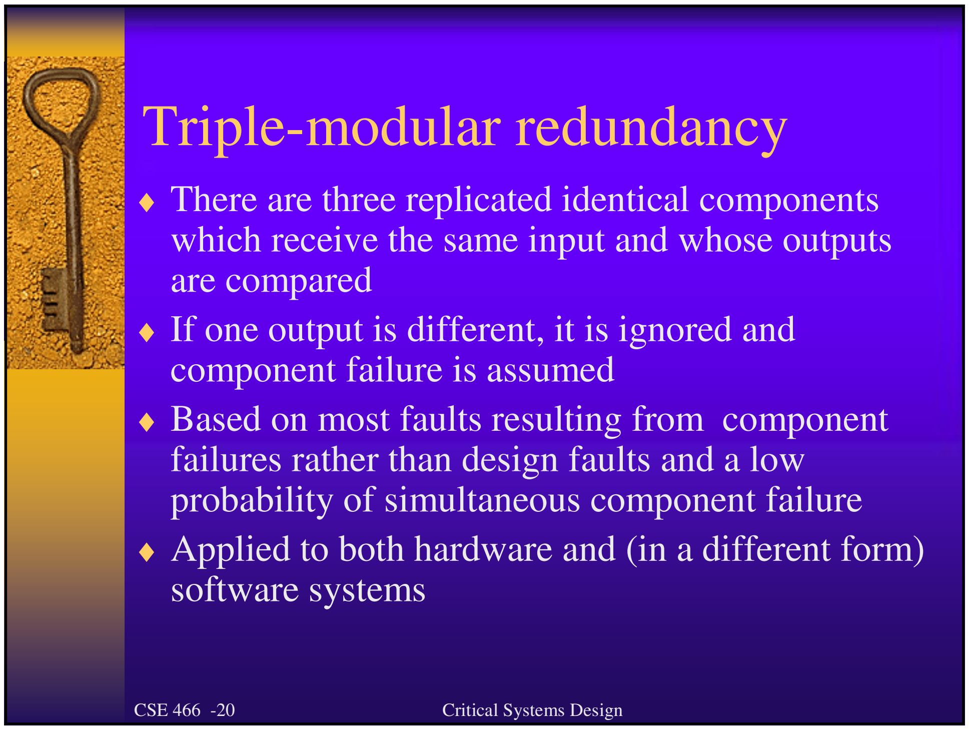
Fault detection

- The use of techniques in the development process which are likely to detect faults before a system is delivered
  - Mathematical correctness arguments
  - Measurement of test coverage
  - Design/program inspections and formal reviews
  - Independent verification and validation
  - Run-time monitoring of the system
  - Back-to-back testing



Fault tolerance

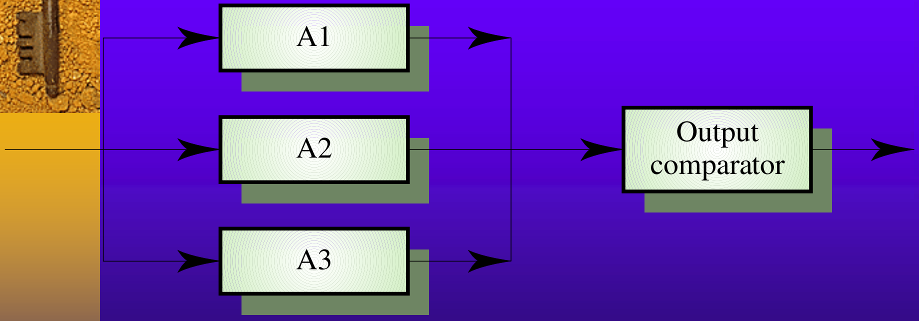
- ♦ In critical situations, systems must be fault tolerant.
- ♦ Fault tolerance means that the system can continue in operation in the presence of system faults
- ♦ Even if the system has been demonstrated to be fault-free, it must also be fault tolerant as there may be specification errors or the validation may be incorrect



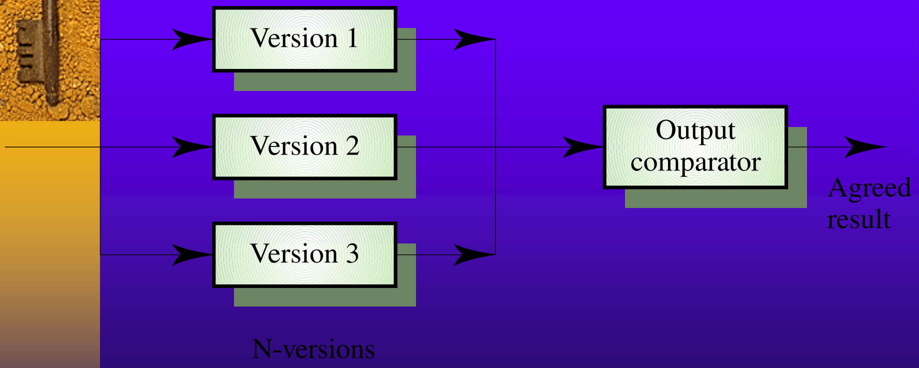
Triple-modular redundancy

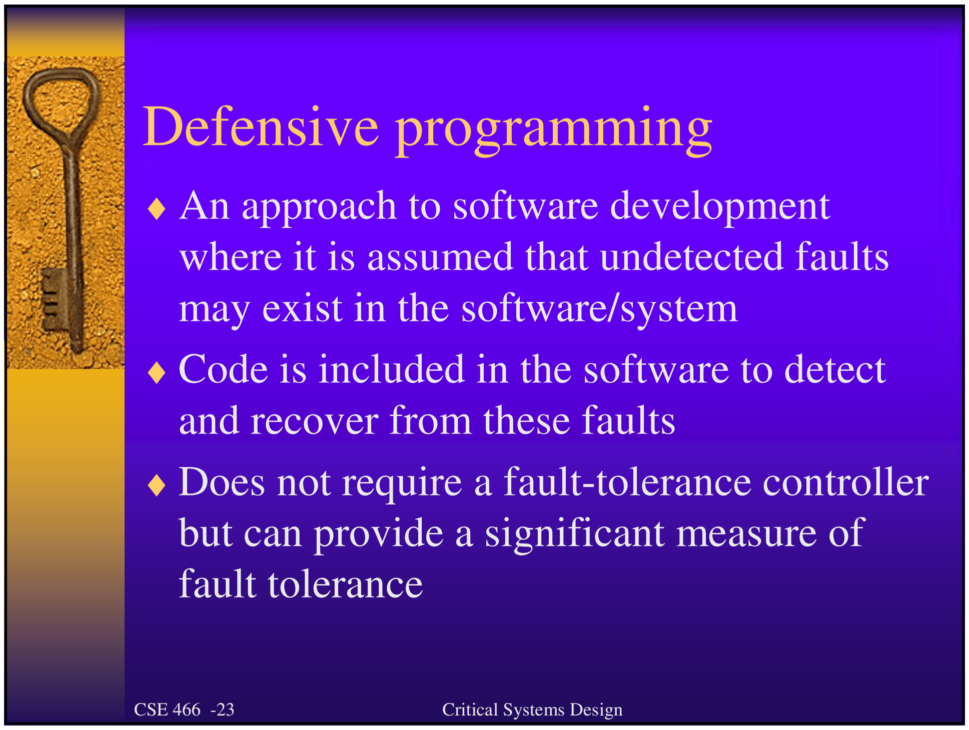
- ♦ There are three replicated identical components which receive the same input and whose outputs are compared
- ♦ If one output is different, it is ignored and component failure is assumed
- ♦ Based on most faults resulting from component failures rather than design faults and a low probability of simultaneous component failure
- ♦ Applied to both hardware and (in a different form) software systems

Hardware reliability with TMR



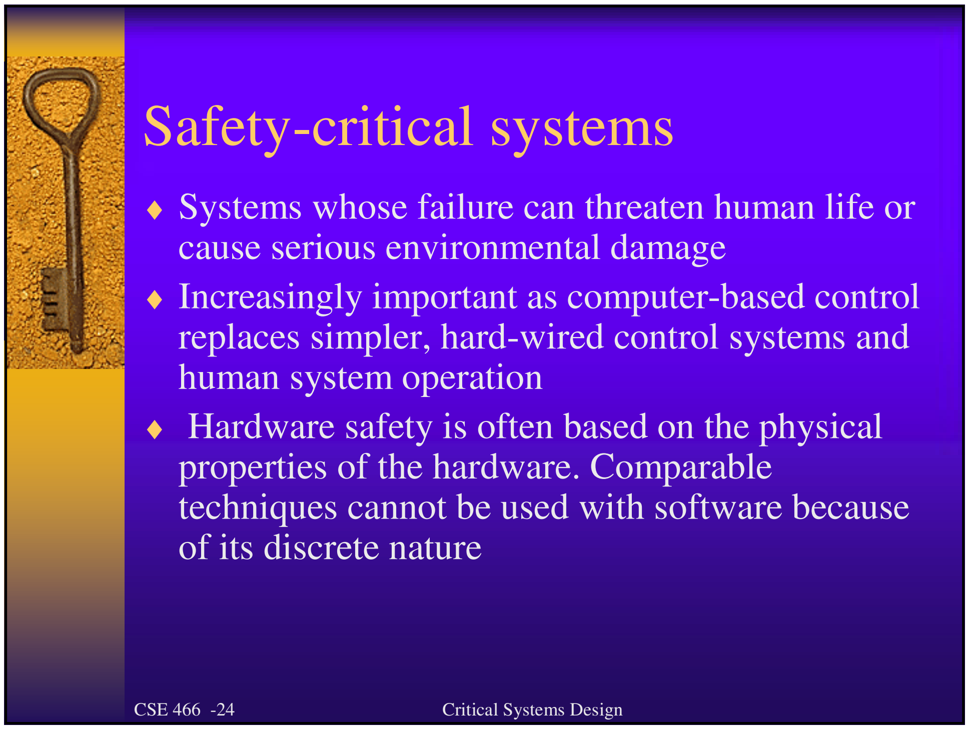
N-version programming





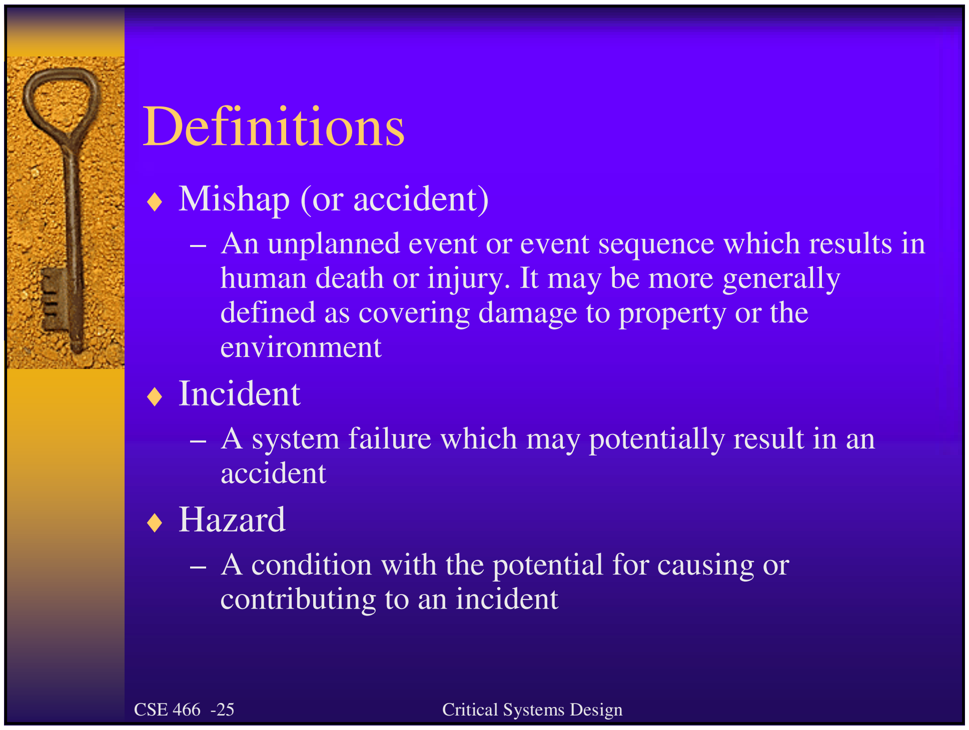
Defensive programming

- An approach to software development where it is assumed that undetected faults may exist in the software/system
- Code is included in the software to detect and recover from these faults
- Does not require a fault-tolerance controller but can provide a significant measure of fault tolerance



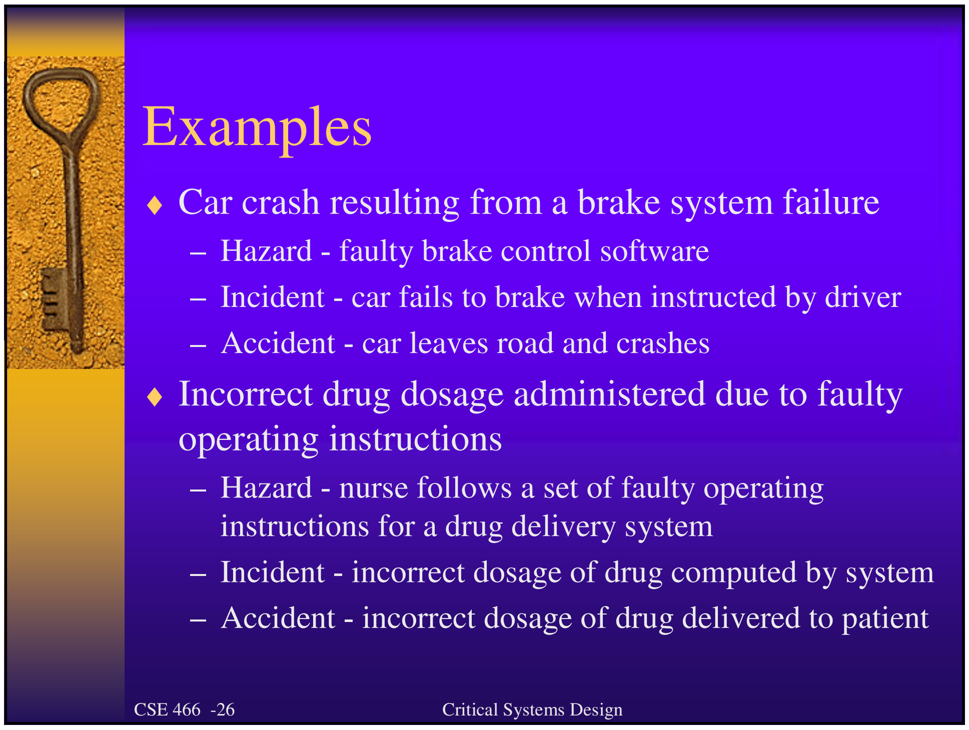
Safety-critical systems

- Systems whose failure can threaten human life or cause serious environmental damage
- Increasingly important as computer-based control replaces simpler, hard-wired control systems and human system operation
- Hardware safety is often based on the physical properties of the hardware. Comparable techniques cannot be used with software because of its discrete nature



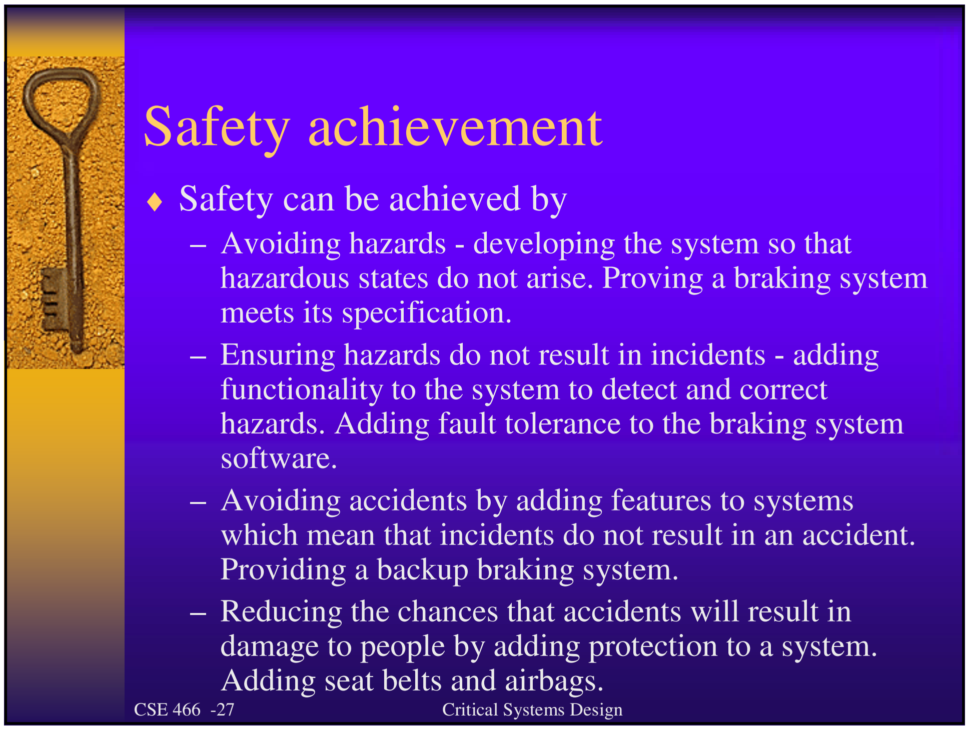
Definitions

- Mishap (or accident)
  - An unplanned event or event sequence which results in human death or injury. It may be more generally defined as covering damage to property or the environment
- Incident
  - A system failure which may potentially result in an accident
- Hazard
  - A condition with the potential for causing or contributing to an incident



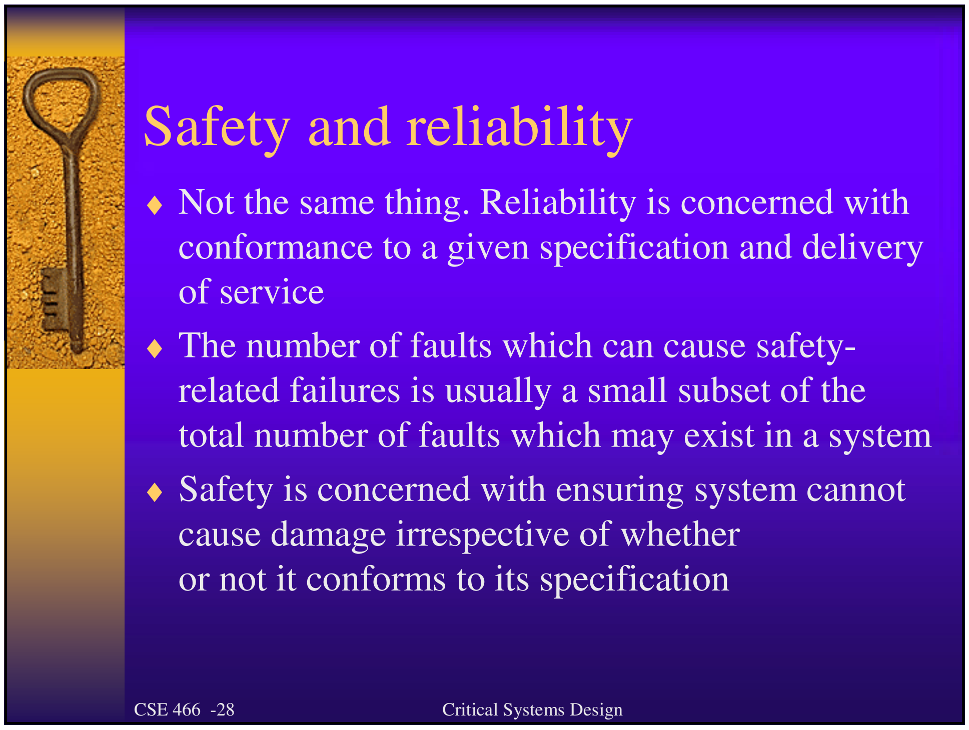
Examples

- Car crash resulting from a brake system failure
  - Hazard - faulty brake control software
  - Incident - car fails to brake when instructed by driver
  - Accident - car leaves road and crashes
- Incorrect drug dosage administered due to faulty operating instructions
  - Hazard - nurse follows a set of faulty operating instructions for a drug delivery system
  - Incident - incorrect dosage of drug computed by system
  - Accident - incorrect dosage of drug delivered to patient



Safety achievement

- Safety can be achieved by
  - Avoiding hazards - developing the system so that hazardous states do not arise. Proving a braking system meets its specification.
  - Ensuring hazards do not result in incidents - adding functionality to the system to detect and correct hazards. Adding fault tolerance to the braking system software.
  - Avoiding accidents by adding features to systems which mean that incidents do not result in an accident. Providing a backup braking system.
  - Reducing the chances that accidents will result in damage to people by adding protection to a system. Adding seat belts and airbags.



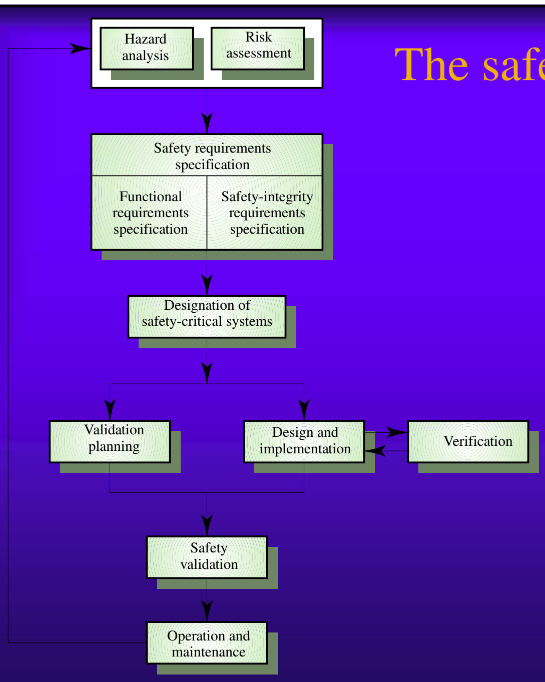
Safety and reliability

- Not the same thing. Reliability is concerned with conformance to a given specification and delivery of service
- The number of faults which can cause safety-related failures is usually a small subset of the total number of faults which may exist in a system
- Safety is concerned with ensuring system cannot cause damage irrespective of whether or not it conforms to its specification

Designing for safety

- System design should always be based around the notion that no single point of failure can compromise system safety. Systems should always be able to tolerate one failure
- However, accidents usually arise because of several simultaneous failures rather than a failure of a single part of the system
- Anticipating complex sub-system interactions when these sub-systems are failing is very difficult

The safety life cycle

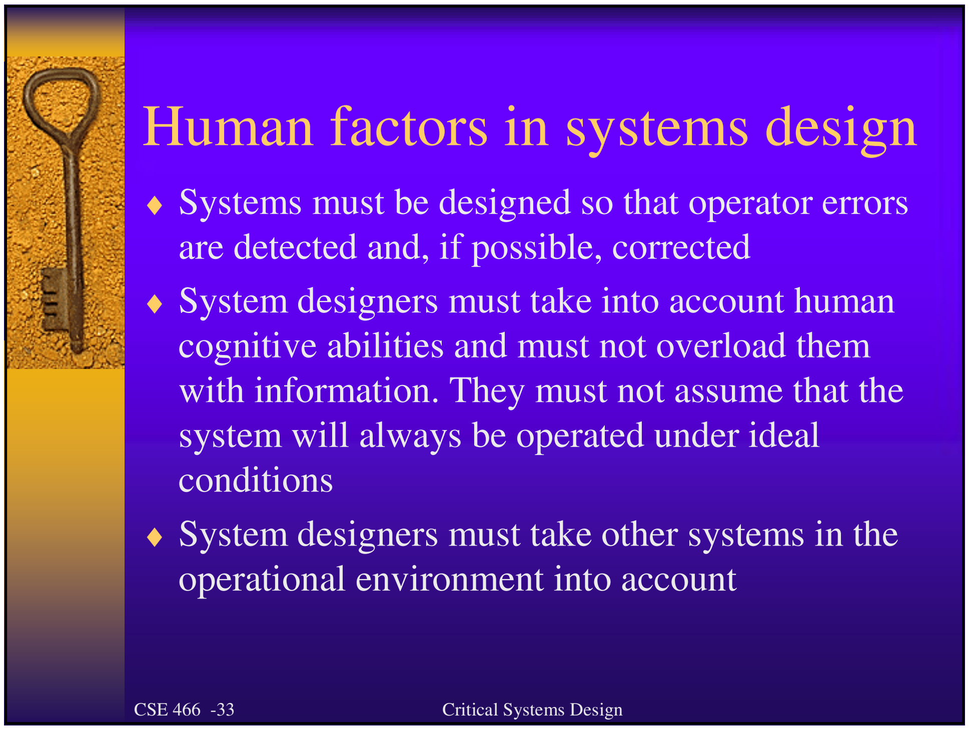

Hazard analysis
Risk assessment
Safety requirements specification
Functional requirements specification
Safety-integrity requirements specification
Designation of safety-critical systems
Validation planning
Design and implementation
Verification
Safety validation
Operation and maintenance

People and critical systems

- ♦ People and associated operational processes are essential elements of critical systems
- ♦ People are probably the most important single source of failure in critical systems BUT they are also the most effective mechanism we have for incident/accident avoidance
- ♦ Human factors are significant in the design, the development and the operation of critical systems

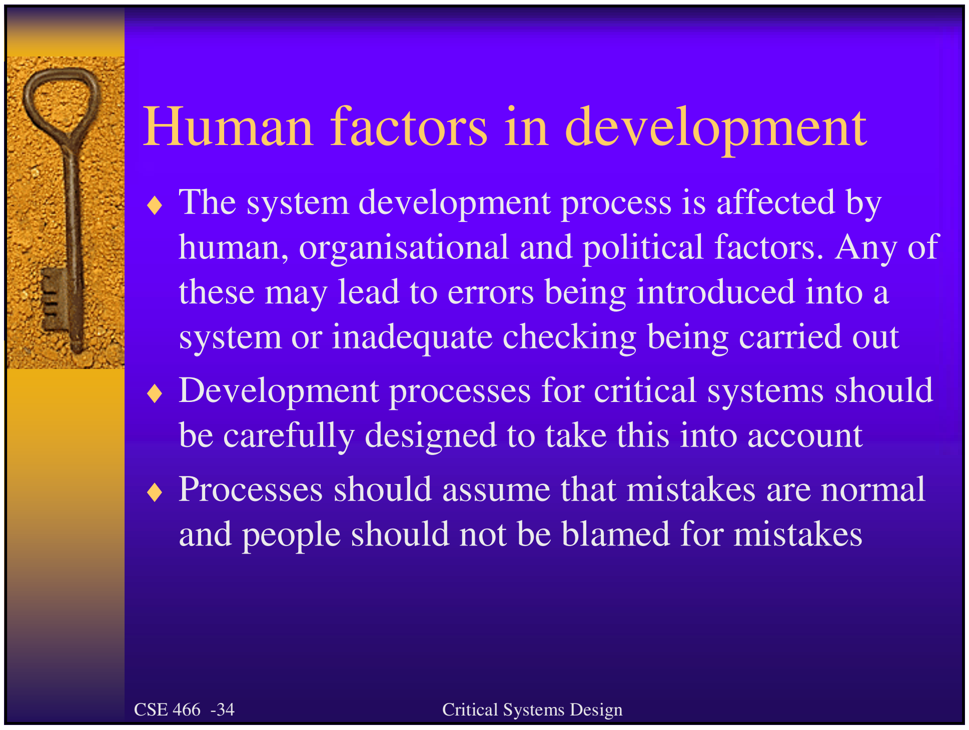
Human factors in operations

- ♦ Many (the majority?) of systems failures are due to ‘errors’ made by operators of the system (pilots, controllers, signallers, etc.)
- ♦ However, it is arguable whether these operators should be blamed for these errors - in many cases they are a result of poor system design where the operational situation was not understood by the system designers



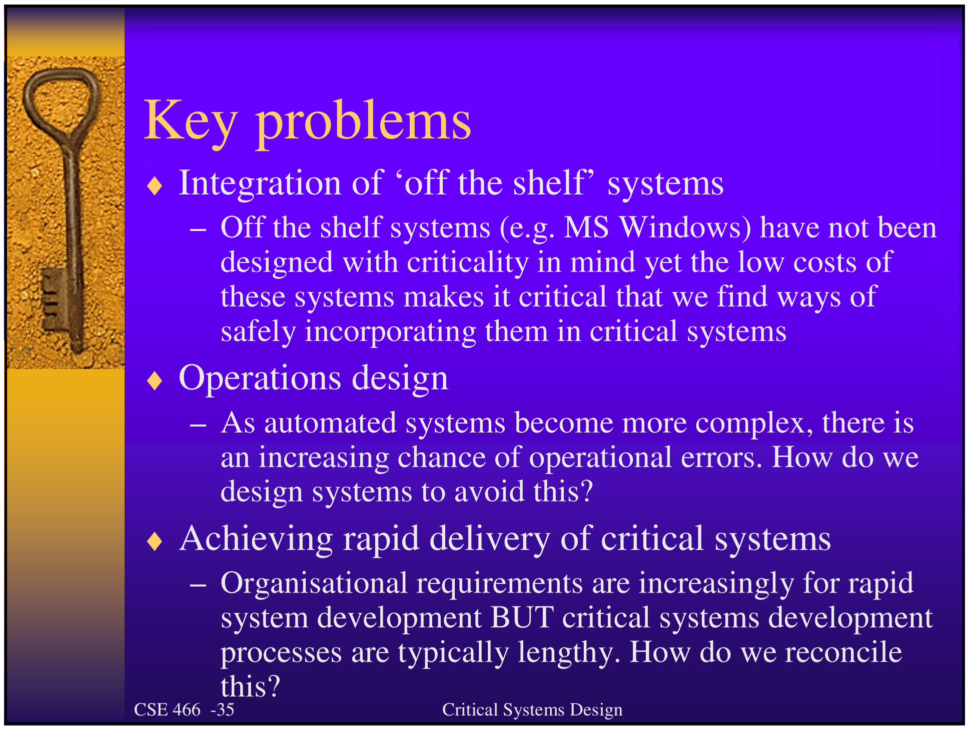
Human factors in systems design

- Systems must be designed so that operator errors are detected and, if possible, corrected
- System designers must take into account human cognitive abilities and must not overload them with information. They must not assume that the system will always be operated under ideal conditions
- System designers must take other systems in the operational environment into account



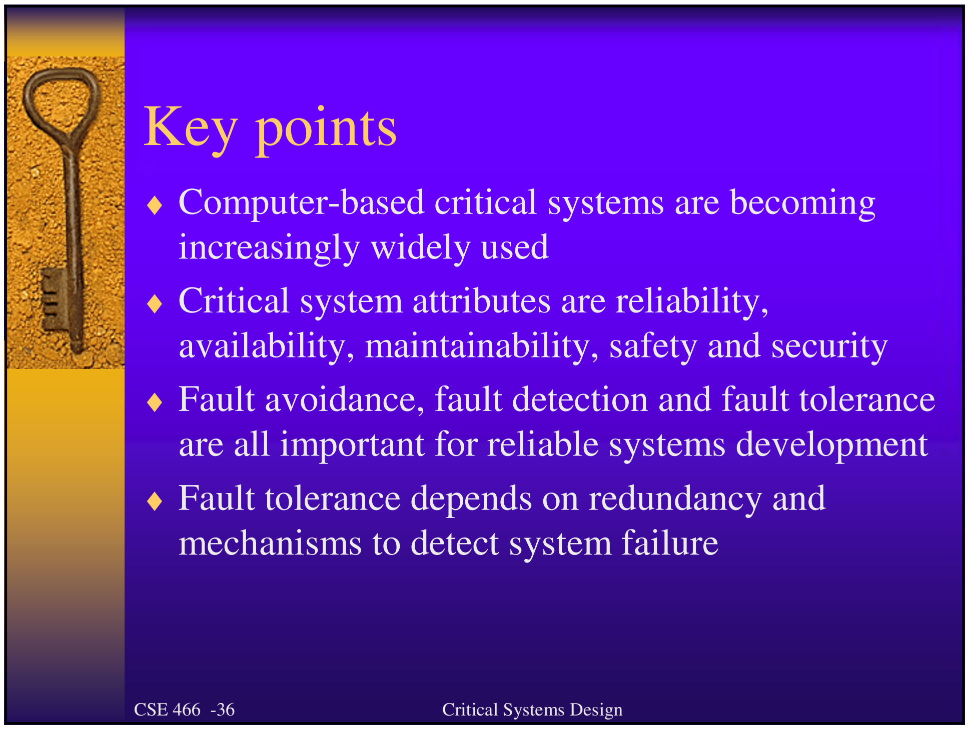
Human factors in development

- The system development process is affected by human, organisational and political factors. Any of these may lead to errors being introduced into a system or inadequate checking being carried out
- Development processes for critical systems should be carefully designed to take this into account
- Processes should assume that mistakes are normal and people should not be blamed for mistakes



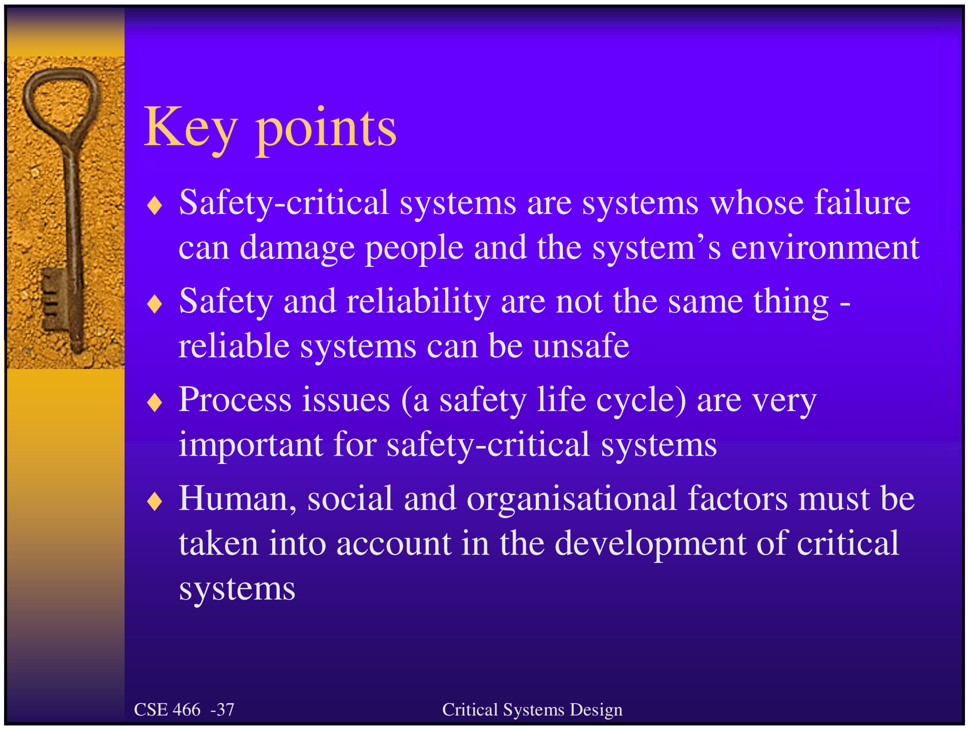
Key problems

- Integration of ‘off the shelf’ systems
  - Off the shelf systems (e.g. MS Windows) have not been designed with criticality in mind yet the low costs of these systems makes it critical that we find ways of safely incorporating them in critical systems
- Operations design
  - As automated systems become more complex, there is an increasing chance of operational errors. How do we design systems to avoid this?
- Achieving rapid delivery of critical systems
  - Organisational requirements are increasingly for rapid system development BUT critical systems development processes are typically lengthy. How do we reconcile this?



Key points

- Computer-based critical systems are becoming increasingly widely used
- Critical system attributes are reliability, availability, maintainability, safety and security
- Fault avoidance, fault detection and fault tolerance are all important for reliable systems development
- Fault tolerance depends on redundancy and mechanisms to detect system failure



Key points

- Safety-critical systems are systems whose failure can damage people and the system's environment
- Safety and reliability are not the same thing - reliable systems can be unsafe
- Process issues (a safety life cycle) are very important for safety-critical systems
- Human, social and organisational factors must be taken into account in the development of critical systems