Critical Systems

- System failure can lead to
  - Loss of life
  - Damage to the environment
  - Loss of much money
- Cost of failure > Cost of the system

Dependable Systems

- Availability - ready for use
- Reliability - works as it should

Good things happen
with some probability

- Safety - does no damage
- Security - resists intrusion

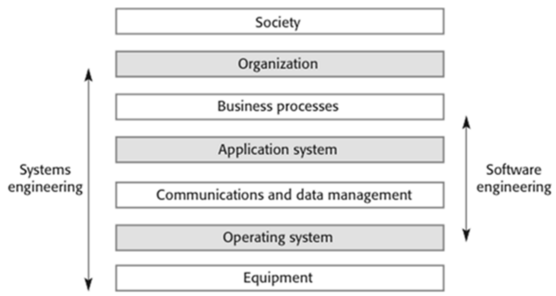
Bad things don't happen
or with very low probability

System perspective

- People
- Software
- Hardware

- Environment

Figure 10.1 The sociotechnical systems stack



Issues

- Requirements
  - How to express dependability (measures)
  - How much is required?
- Design, Implementation
  - How to make the system dependable?
- Verification
  - How to verify dependability

	Terminology	Specification	Design & Impl.	Verification
Avail & Reliab.	11.2	12.3	13	15.2
Safety	11.3	12.2/5		15.1/4/5
Secure	11.4	12.4	14	15.3

Sommerville 9th edition

Terminology

Fault	- static: undesirable state
Failure	- dynamic: undesirable behaviour
Hazard	- situation out of control
Accident	- event(s) causing damage
Damage	- resulting loss

Reliability metrics [12.3.1]

- AVAIL probability that system is available
- POFOD Probability Of Failure On Demand
  - irregular use: fire alarm
- ROCOF Rate of OCcurrence Of Failure
  - regular use: coffee machine
  - per time unit (week) or usage (per 1000 cups)
- MTTF Mean Time To Failure
  - long transactions (editor)
    = 1/ROCOF

Specify per failure!

- Planned / unplanned unavailability
- Transient / Permanent (requires service)
- Corrupting data?

Reliability costs - be realistic!

- POFOD
  - can be quite high: 1/100 - 1/1000
- AVAIL
  - 99% - 14 minutes/day
  - 99.9% - 10 minutes/week
  - 99.99% - 1 minute/week
  - 99.999% - 5 minutes/year
- Low probabilities cannot be tested!

Reliability testing [15.2]

- Statistical testing
  - does not work for very high reliability
  - fault injection

	unknown faults	injected faults
found		
not found	?	

Safety Terminology

Fault		
Failure	Hazard probability	
Hazard		Risk
Accident	Hazard severity	
Damage		

Risk-driven analysis

- Hazard identification
  – what are the hazards?
- Risk assessment
  – risk from this hazard is acceptable?
- Hazard analysis
  – how does the hazard occur?
- Risk reduction

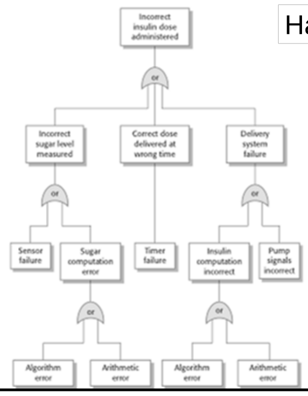
Domain knowledge

Figure 12.3 Risk classification for the insulin pump

Identified hazard	Hazard probability	Accident severity	Estimated risk	Acceptability
1.Insulin overdose computation	Medium	High	High	Intolerable
2. Insulin underdose computation	Medium	Low	Low	Acceptable
3. Failure of hardware monitoring system	Medium	Medium	Low	ALARP
4. Power failure	High	Low	Low	Acceptable
5. Machine incorrectly fitted	High	High	High	Intolerable
6. Machine breaks in patient	Low	High	Medium	ALARP
7. Machine causes infection	Medium	Medium	Medium	ALARP
8. Electrical interference	Low	High	Medium	ALARP
9. Allergic reaction	Low	Low	Low	Acceptable

Figure 12.4 An example of a fault tree

Hazard analysis



Risk reduction

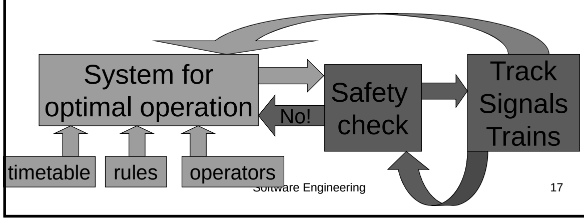
Fault
Failure
Hazard
Accident
Damage

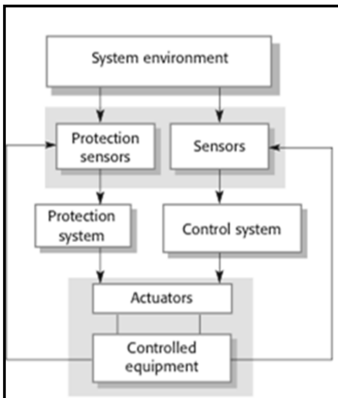
Fault avoidance / correction
Fault tolerance
Hazard detection

Damage limitation

Design principles

- Minimize the critical part
  – cost increases much with size and safety
- Example (railway operation)





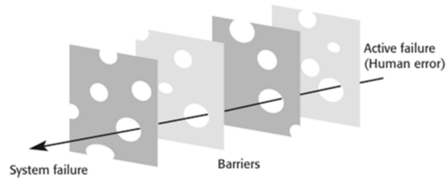
Protection system architecture

Fault tolerance

- Fault $\not\longrightarrow$ Failure
- Hardware faults
  - works … fails
- Software faults
  - present from the start
- Human error

Human error [10.5.1]

- Humans *will* make errors
  - the system needs barriers



Human error

- The human made the error because ...
  - lack of information
  - information overload
  - badly designed user interface
  - ”official” routines are not practical
  - pressure to take ”shortcuts”
  - inadequate training / practice

Technology

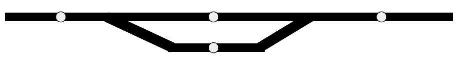
Organisation

Fault tolerance

- Detect fault
- Avoid failure
  - go into **<u>safe state</u>**
  - less functionality
  - railway example: all signals red
  - traffic light example: blinking yellow
- Make sure fault is noticed

Safe state design

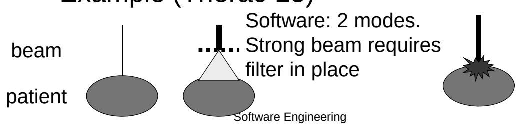
- Example: railway track indicator lamp



  - Lamp on = track is free
  - Lamp off = train detected
- Why?
  - What if the lamp fails ...

Safety devices

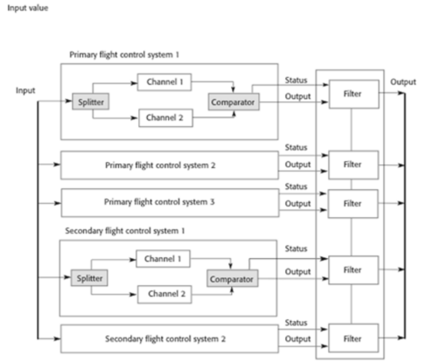
- Simple
- Preferably in hardware
- Preferably autonomous
  - depend on gravity, not electricity
- Example (Therac-25)



Fault tolerance techniques

- Redundancy (spare components)
  - best for hardware
  - for safety, availability
- Diversity (*different* components)
  - design errors (SW, HW)
  - different hardware, supplier, software
  - simpler secondary system
- Monitoring

Figure 13.5 Airbus flight control system architecture



Software diversity

- N-version programming
- Diversity in
  - design method
  - programming language
  - tools
- Problem: specification errors
  - formal specification + verification

Fault avoidance

- Formal development
- Dependable programming [13.4]
  - Hiding, ADT, OOP
  - Name all constants
  - Check inputs, array bounds
  - Exception handling
  - Timeouts, restarts, rollbacks

- Minimize error-prone constructs
  - pointers - pointer arithmetic
    - impossible to verify too
  - dynamic memory management
    - stack overflow (recursion)
  - floating point numbers
    - beware of integer overflow too
  - aliasing, inheritance (name - object)
  - parallelism, interrupts

Fault detection and correction

- Cannot test ”shall not” requirements
- Formal verification [15.1]
  - Model checking
  - Correctnes proofs
- Safety cases [15.5]
  - Structured argument: ”this cannot happen”
  - Producing the argument reveals: ”it can happen” = fault detection

Process [13.2, 15.4]

- Standardized process
  - Precise specification
    - Assign Safety Integrity Level (SIL)
  - Safety reviews (hazard monitoring)
  - Diverse verification
    (inspection, model checking, test, proof)
  - Version management
- Quality culture (process is accepted)

Process

- Documentation
  - Auditable
- Independent safety regulator
  - process is dependable
    - do we have the right process
  - process is followed
    - are we doing the process right