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
- Safety - does no damage
- Security - resists intrusion

System perspective

- People
- Software
- Hardware
  - Environment

Issues

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

Figure 10.1 The sociotechnical systems stack

<table>
<thead>
<tr>
<th>Terminology</th>
<th>Specification</th>
<th>Design &amp; Impl.</th>
<th>Verification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avail &amp; Reliab.</td>
<td>11.2</td>
<td>12.3</td>
<td>13</td>
</tr>
<tr>
<td>Safety</td>
<td>11.3</td>
<td>12.2/5</td>
<td>15.1/4/5</td>
</tr>
<tr>
<td>Secure</td>
<td>11.4</td>
<td>12.4</td>
<td>14</td>
</tr>
</tbody>
</table>

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)  
  - $= 1/\text{ROCOF}$

**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

<table>
<thead>
<tr>
<th>found</th>
<th></th>
<th>not found</th>
</tr>
</thead>
<tbody>
<tr>
<td>unknown faults</td>
<td>injected faults</td>
<td>?</td>
</tr>
</tbody>
</table>

**Safety Terminology**

- Fault  
- Failure  
- Hazard  
- Accident  
- Damage  
- Hazard probability  
- Hazard severity  
- Risk
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

Risk classification for the insulin pump

<table>
<thead>
<tr>
<th>Identified hazard</th>
<th>Hazard probability</th>
<th>Accident severity</th>
<th>Estimated risk</th>
<th>Acceptability</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Insulin overdose computation</td>
<td>Medium</td>
<td>High</td>
<td>High</td>
<td>Intolerable</td>
</tr>
<tr>
<td>2. Insulin underdose computation</td>
<td>Medium</td>
<td>Low</td>
<td>Low</td>
<td>Acceptable</td>
</tr>
<tr>
<td>3. Failure of hardware monitoring system</td>
<td>Medium</td>
<td>Medium</td>
<td>Low</td>
<td>ALARP</td>
</tr>
<tr>
<td>4. Power failure</td>
<td>High</td>
<td>Low</td>
<td>Low</td>
<td>Acceptable</td>
</tr>
<tr>
<td>5. Machine incorrectly filled</td>
<td>High</td>
<td>High</td>
<td>High</td>
<td>Intolerable</td>
</tr>
<tr>
<td>6. Machine breaks in patient</td>
<td>Low</td>
<td>High</td>
<td>Medium</td>
<td>ALARP</td>
</tr>
<tr>
<td>7. Machine causes interaction</td>
<td>Medium</td>
<td>Medium</td>
<td>Medium</td>
<td>ALARP</td>
</tr>
<tr>
<td>8. Electrical interference</td>
<td>Low</td>
<td>High</td>
<td>Medium</td>
<td>ALARP</td>
</tr>
<tr>
<td>9. Allergen reaction</td>
<td>Low</td>
<td>Low</td>
<td>Low</td>
<td>Acceptable</td>
</tr>
</tbody>
</table>

Design principles

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

Risk reduction

Fault avoidance / correction
- Fault tolerance
- Hazard detection
- Damage limitation

Protection system architecture

System environment
- Protection sensors
- Sensors
- Protection system
- Control system
- Actuators
- Controlled equipment

Protection

Safety check

System for optimal operation
- Timetable
- Rules
- Operators

Track signals
- Trains
Fault tolerance
- Fault $\rightarrow$ 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

Fault tolerance
- Detect fault
- Avoid failure
  - go into safe state
  - 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)
  - Software: 2 modes.
  - Strong beam requires filter in place
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

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

Fault detection and correction

• Cannot test "shall not" requirements
• Formal verification [15.1]
  – Model checking
  – Correctness 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