Chapter 12 – Safety Engineering

Topics covered

✧ Safety-critical systems
✧ Safety requirements
✧ Safety engineering processes
✧ Safety cases

Safety

✧ A property of a system
✧ The system's ability to operate services
- Prevent danger causing human injury or death
- Avoiding damage to the system's environment.

✧ Software safety issues become important
- Most devices incorporate software-based control systems.
- Control real-time, safety-critical processes.

Software in safety-critical systems

✧ Software-controlled systems
- Decisions are made by the software.
- Subsequent actions are safety-critical.
- Software behaviour is related to safety of the system.

✧ Checking and monitoring safety-critical components
- E.g., monitoring aircraft engine components for fault detection.

✧ Monitoring software is safety-critical
- Other components may fail due to the failure of fault detection.

Safety and reliability

✧ Safety and reliability
- Reliability and availability are not sufficient for system safety

✧ Reliability
- Conformance to a given specification and delivery of service

✧ Safety
- Ensuring system cannot cause damage.

✧ System reliability is essential for safety
- However, reliable systems can be unsafe

Unsafe reliable systems

✧ Dormant system faults
- Undetected for a number of years and only rarely arise.

✧ Specification errors
- Software system behaves as specified but cause an accident.

✧ Hardware failures at runtime
- E.g., generating spurious inputs
- Hard to anticipate in the specification

✧ Context-sensitive commands
- E.g., a system command is executed at the wrong time.

Safety-critical systems

Safety critical systems

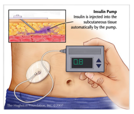
- ✧ Essential that system operation is always safe
  - Must not cause damage to people or the system's environment
- ✧ Examples
  - Process control systems in chemical manufacture
  - Automobile control systems such as braking management systems

Safety criticality

- ✧ Primary safety-critical systems
  - Embedded software systems
  - Cause associated hardware failures, directly threatening people.
  - E.g., the insulin pump control system.
- ✧ Secondary safety-critical systems
  - Result in faults in other connected systems, affecting safety consequences.
  - E.g., the Mentcare system producing inappropriate treatment being prescribed.
  - Infrastructure control systems.

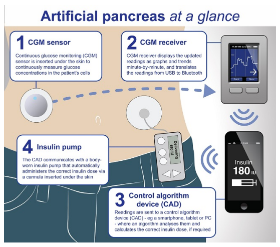
Insulin pump control system

- ✧ Collects data from a blood sugar sensor and calculates the amount of insulin required to be injected.
- ✧ Calculation based on the rate of change of blood sugar levels.


Insulin Pump
Insulin is injected into the subcutaneous tissue automatically by the pump.
0.8
The Hughston Foundation, Inc. ©2007

Insulin pump control system (cont.)

- ✧ Sends signals to a micro-pump to deliver the correct dose of insulin.
- ✧ Safety-critical system as low blood sugars can lead to brain malfunctioning, coma and death; high-blood sugar levels have long-term consequences such as eye and kidney damage.


Artificial pancreas *at a glance*
1 CGM sensor
Continuous glucose monitoring (CGM) sensor is inserted under the skin to continuously measure glucose concentrations in the patient's cells
2 CGM receiver
CGM receiver displays the updated readings as graphs and trends minute-by-minute, and translates the readings from USB to Bluetooth
3 Control algorithm device (CAD)
Readings are sent to a control algorithm device (CAD) - eg a smartphone, tablet or PC - where an algorithm analyses them and calculates the correct insulin dose, if required
4 Insulin pump
The CAD communicates with a body-worn insulin pump that automatically administers the correct insulin dose via a cannula inserted under the skin
Delivering 180 IU
Insulin 180 IU

Safety criticality

- ✧ Primary safety-critical systems
  - Embedded software systems
  - Cause associated hardware failures, directly threatening people.
  - E.g., the insulin pump control system.
- ✧ Secondary safety-critical systems
  - Result in faults in other connected systems, affecting safety consequences.
  - E.g., the Mentcare system producing inappropriate treatment being prescribed.
  - Infrastructure control systems.

Hazards

✧ Situations or events that can lead to an accident
- Incorrect computation by software in navigation system
- Failure to detect possible disease in medication prescribing system

✧ Perform accident prevention actions
- Hazards do not inevitably lead to accidents

Safety achievement

✧ Hazard avoidance
- Appling hazard avoidance design to software systems.
- Prevent some classes of hazard.

✧ Hazard detection and removal
- Detecting and removing hazard before causing accidents.

✧ Damage limitation
- Protection features to minimise the damage.

Safety terminology

Term	Definition
Accident (or mishap)	An unplanned event or sequence of events which results in human death or injury, damage to property, or to the environment. An overdose of insulin is an example of an accident.
Hazard	A condition with the potential for causing or contributing to an accident. A failure of the sensor that measures blood glucose is an example of a hazard.
Damage	A measure of the loss resulting from a mishap. Damage can range from many people being killed as a result of an accident to minor injury or property damage. Damage resulting from an overdose of insulin could be serious injury or the death of the user of the insulin pump.
Hazard severity	An assessment of the worst possible damage that could result from a particular hazard. Hazard severity can range from catastrophic, where many people are killed, to minor, where only minor damage results. When an individual death is a possibility, a reasonable assessment of hazard severity is 'very high'.
Hazard probability	The probability of the events occurring which create a hazard. Probability values tend to be arbitrary but range from 'probable' (say 1/100 chance of a hazard occurring) to 'implausible' (no conceivable situations are likely in which the hazard could occur). The probability of a sensor failure in the insulin pump that results in an overdose is probably low.
Risk	This is a measure of the probability that the system will cause an accident. The risk is assessed by considering the hazard probability, the hazard severity, and the probability that the hazard will lead to an accident. The risk of an insulin overdose is probably medium to low.

Normal accidents

✧ Rarely have a single cause in complex systems

✧ Designed to be resilient to a single point of failure

✧ A fundamental principle of safe systems design
- A single point of failure does not cause an accident.

✧ A result of combinations of malfunctions.

✧ Hard to anticipate all combinations in software systems
- Difficult to achieve complete safety.
- Accidents are inevitable.

Software safety benefits

✧ Software control systems contributes to system safety
- A large number of conditions to be monitored and controlled.
- Reducing human efforts and time in hazardous environments.
- Detecting and repairing safety-critical operator errors.

Safety requirements

Functional and non-functional requirements

✧ Functional requirements
- Statements of services the system should provide,
- How the system should react to particular inputs and how the system should behave in particular situations.

✧ Non-functional requirements
- Constraints on the services or functions of the system.
- Apply to the whole system rather than individual features.

Functional requirements

✧ Describe functionality or system services.
- Depending on the type of software systems and users.

✧ Functional user requirements
- High-level statements of what the system should do.

✧ Functional system requirements
- The system services in detail.

Safety specification

✧ Goal
- Identifying protection requirements.
- Preventing injury or death or environmental damage.

✧ Safety requirements
- Shall Not requirements.
- Define situations and events that should never occur.

✧ Functional safety requirements
- Checking and recovery features in a system.
- Protection feature against failures and external attacks.

Hazard-driven analysis

✧ Hazard identification

✧ Hazard assessment

✧ Hazard analysis

✧ Risk reduction
- Safety requirements specification

Hazard identification

✧ Identify the hazards threatening the system.

✧ Different types of hazard:
- Physical hazards
- Electrical hazards
- Biological hazards
- Service failure hazards
- Etc.

Insulin pump risks

✧ Insulin overdose (service failure).

✧ Insulin underdose (service failure).

✧ Power failure due to exhausted battery (electrical).

✧ Electrical interference with other medical equipment (electrical).

✧ Poor sensor and actuator contact (physical).

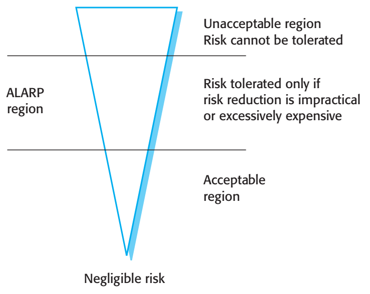
✧ Infection caused by introduction of machine (biological).

✧ Allergic reaction to materials or insulin (biological).

Hazard assessment

- ✧ Understanding the likelihood that a risk will arise and the potential consequences.
- ✧ Risks category:
- ✧ Intolerable.
  - Unsupportable.
- ✧ As low as reasonably practical (ALARP).
  - Minimising risk possibilities given available resources.
- ✧ Acceptable.
  - No extra costs to reduce hazard probability.

The risk triangle



Social acceptability of risk

- ✧ The acceptability of a risk.
  - Human, social and political considerations.
- ✧ Society is less willing to accept risk in most cases.
  - E.g., the costs of cleaning up or preventing pollution.
- ✧ Subjective assessment
  - Depending on evaluators making the assessment.

Hazard assessment

- ✧ The risk probability and the risk severity.
- ✧ Relative values: ‘unlikely’, ‘rare’, ‘very high’, etc.
  - Impossible to do precise measurement
- ✧ Goal:
  - Prevent or remove potential risks with the high severity.

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

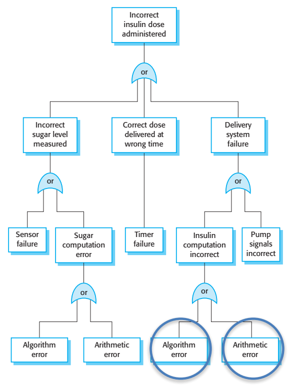
Hazard analysis

- ✧ The root causes of risks in a particular system.
- ✧ Hazard analysis techniques
  - Inductive, bottom-up techniques:
    Evaluate the hazards, starting with system failures.
  - Deductive, top-down techniques:
    Reason failure causes, starting with a hazard

Fault-tree analysis

- ✧ A deductive top-down technique.
- ✧ Hazard at the root of the tree
  - Identify states causing hazards.
- ✧ Linking conditions by relationships (e.g., ‘and’ or ‘or’)
- ✧ Goal
  - Minimizing the number of single failure causes.

An example of a software fault tree



Fault tree analysis

- ✧ Possible conditions of incorrect dose of insulin:
  - Incorrect measurement of blood sugar level
  - Failure of delivery system
  - Dose delivered at wrong time
- ✧ Root causes of these hazards:
  - Algorithm error
  - Arithmetic error

Risk reduction

- ✧ Goal:
  - Identify requirements for risk managements to avoid accidents.
- ✧ Risk reduction strategies
  - Hazard avoidance
  - Hazard detection and removal
  - Damage limitation

Strategy use

- ✧ Combining multiple risk reduction strategies
- ✧ E.g., a chemical plant control system:
  - Detecting and correcting excess pressure in the reactor.
  - Opening a relief valve as independent protection system

Insulin pump - software risks

- ✧ Arithmetic error
  - Data variable overflow or underflow during a computation.
  - Handing runtime exception.
- ✧ Algorithmic error
  - Comparison between previous and current values
  - Checking the maximum value to control dose.

Examples of safety requirements

SR1: The system shall not deliver a single dose of insulin that is greater than a specified maximum dose for a system user.

SR2: The system shall not deliver a daily cumulative dose of insulin that is greater than a specified maximum daily dose for a system user.

SR3: The system shall include a hardware diagnostic facility that shall be executed at least four times per hour.

SR4: The system shall include an exception handler for all of the exceptions that are identified in Table 3.

SR5: The audible alarm shall be sounded when any hardware or software anomaly is discovered and a diagnostic message, as defined in Table 4, shall be displayed.

SR6: In the event of an alarm, insulin delivery shall be suspended until the user has reset the system and cleared the alarm.

Safety engineering processes

Chapter 12 – Safety Engineering

- ~~✧ Safety-critical systems~~
- ~~✧ Safety requirements~~
- ✧ Safety engineering processes
- ✧ Safety cases

Safety engineering processes

Safety engineering processes

- ✧ Reliability engineering processes
  - Reviews and checks at each stage in the process
  - General goal of fault avoidance and fault detection
  - Safety reviews and explicit identification of hazards

Regulation

- ✧ Evidence that safety engineering processes used.
- ✧ For example:
  - The specification and records of the checks.
  - Evidence of the verification and validation the results.
  - Organizations for dependable software processes.

Agile methods and safety

- ✧ Agile methods are not usually used for safety-critical systems engineering
  - Extensive documentation.
  - A detailed safety analysis of a complete system specification.
- ✧ Test-driven development may be used

Safety assurance processes

- ✧ Defining and ensuring a dependable process.
- ✧ Process assurance focuses on:
  - The processes are appropriate for dependability required.
  - The processes are followed by the development team.
- ✧ Should generate documentation
  - Agile processes are not used for critical systems.

Processes for safety assurance

- ✧ Process assurance is important for safety-critical systems development:
  - Testing may not find all problems.
- ✧ Safety assurance activities
  - Record the analyses.
  - Personal responsibility.

Safety related process activities

- ✧ A hazard logging and monitoring system.
- ✧ Safety engineers who responsible for safety.
- ✧ Extensive use of safety reviews.
- ✧ A safety certification system.
- ✧ Detailed configuration management

Hazard analysis

- ✧ Identifying hazards and their root causes.
- ✧ Traceability from identified hazards
  - Analysis to to ensure that the hazards have been covered.
- ✧ A hazard log may be used to track hazards.

Safety reviews

- ✧ Driven by the hazard register.
- ✧ Assess the system and judge whether to cope with hazards in a safe way.

Formal verification

- ✧ A mathematical specification of the system is produced.
- ✧ Static verification technique used in development:
  - A formal specification -- mathematically analyzed for consistency.
    - Discover specification errors and omissions.
  - A program conforms to its mathematical specification
    - Programming and design errors.

Arguments for formal methods

- ✧ A mathematical specification requires a detailed analysis
- ✧ Concurrent systems
  - Discover race conditions.
  - Testing is difficult.
- ✧ Detect implementation errors before testing
  - Program is analyzed alongside the specification.

Arguments against formal methods

- ✧ Require specialized notations
  - Cannot be understood by domain experts.
- ✧ Expensive to develop a specification
- ✧ Proofs may contain errors.
- ✧ More cheaply using other V & V techniques.
- ✧ The proof making incorrect assumptions
  - System's behavior lies outside the scope of the proof.

Model checking

- ✧ Create a state model of using a specialized system
  - Checking the model for errors.
- ✧ The model checker explores all possible paths
  - Checks that a user-specified property is valid for each path.
- ✧ Verifying concurrent systems, which are hard to test.
- ✧ Model checking is computationally very expensive
  - Verification of small to medium sized critical systems.

Static program analysis

- ✧ Tools for source text processing.
- ✧ Parse the program text to discover erroneous conditions.
- ✧ Effective as an aid to inspections
  - A supplement to inspections.

Automated static analysis checks

Fault class	Static analysis check
Data faults	Variables used before initialization Variables declared but never used Variables assigned twice but never used between assignments Possible array bound violations Undeclared variables
Control faults	Unreachable code Unconditional branches into loops
Input/output faults	Variables output twice with no intervening assignment
Interface faults	Parameter-type mismatches Parameter number mismatches Non-usage of the results of functions Uncalled functions and procedures
Storage management faults	Unassigned pointers Pointer arithmetic Memory leaks

Levels of static analysis

- ✧ Characteristic error checking
  - Check for patterns in the code for characteristic of errors.
- ✧ User-defined error checking
  - Define error patterns, extending error types by specific rules
- ✧ Assertion checking
  - Formal assertions in their program
  - Symbolically executes to find potential problems.

Example for Symbolic Execution

```
x = readInput();
y = x * 2;

if (y == 12) {
  fail();
} else {
  print("OK");
}
```

Use of static analysis

- ✧ Particularly valuable when a language has weak typing
  - Many errors are undetected by the compiler.
- ✧ Security checking
  - Discover areas of vulnerability such as buffer overflows.
- ✧ The development of safety and security critical systems.

Safety cases

Sep 19

Safety and dependability cases

- ✧ Safety and dependability cases
  - Structured documents
  - Evidence of a required level of safety or dependability.
- ✧ Regulators check a system is as safe or dependable.
- ✧ Regulators and developers work together for a system safety/dependability case.

The system safety case

- ✧ A safety case is:
  - A documented body of evidence.
  - A system is adequately safe for a given environment.
- ✧ Formal proof, design rationale, safety proofs, etc.
- ✧ Wider system safety case that takes hardware and operational issues into account.

The contents of a software safety case

Chapter	Description
System description	An overview of the system and a description of its critical components.
Safety requirements	The safety requirements abstracted from the system requirements specification. Details of other relevant system requirements may also be included.
Hazard and risk analysis	Documents describing the hazards and risks that have been identified and the measures taken to reduce risk. Hazard analyses and hazard logs.
Design analysis	A set of structured arguments that justify why the design is safe.
Verification and validation	A description of the V & V procedures used and, where appropriate, the test plans for the system. Summaries of the test results showing defects that have been detected and corrected. If formal methods have been used, a formal system specification and any analyses of that specification. Records of static analyses of the source code.

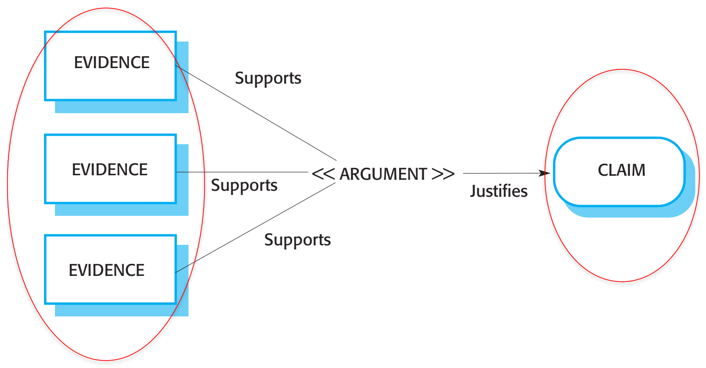
Chapter	Description
Review reports	Records of all design and safety reviews.
Team competences	Evidence of the competence of all of the team involved in safety-related systems development and validation.
Process QA	Records of the quality assurance processes (see Chapter 24) carried out during system development.
Change management processes	Records of all changes proposed, actions taken and, where appropriate, justification of the safety of these changes. Information about configuration management procedures and configuration management logs.
Associated safety cases	References to other safety cases that may impact the safety case.

Structured arguments

✧ Safety cases be based on structured arguments

✧ Claims of safety and security justified by evidences.

Structured arguments



Insulin pump safety argument

✧ Arguments are based on claims and evidence.

✧ Insulin pump safety:

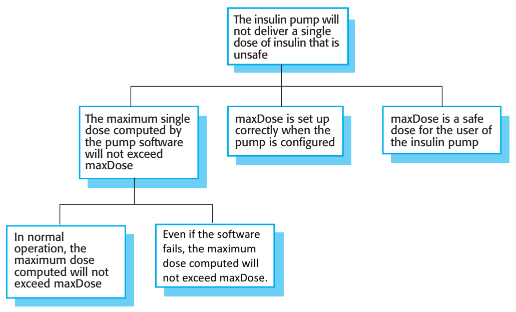
- Claim: The maximum single dose of insulin to be delivered (CurrentDose) will not exceed MaxDose.
- Evidence: Safety argument for insulin pump
- Evidence: Test data for insulin pump. The value of currentDose was correctly computed in 400 tests
- Evidence: Static analysis report for insulin pump software revealed no anomalies that affected the value of CurrentDose
- Argument: The evidence presented demonstrates that the maximum dose of insulin that can be computed = MaxDose.

Structured safety arguments

✧ Structured arguments of a system safety obligations.

✧ Generally based on a claim hierarchy.

A safety claim hierarchy for the insulin pump


The insulin pump will not deliver a single dose of insulin that is unsafe
The maximum single dose computed by the pump software will not exceed maxDose
maxDose is set up correctly when the pump is configured
maxDose is a safe dose for the user of the insulin pump
In normal operation, the maximum dose computed will not exceed maxDose
Even if the software fails, the maximum dose computed will not exceed maxDose.

Software safety arguments

✧ Show that the system cannot reach in unsafe state.

✧ These are weaker than correctness arguments which must show that the system code conforms to its specification.

✧ They are generally based on proof by contradiction

- Assume that an unsafe state can be reached;
- Show that this is contradicted by the program code.

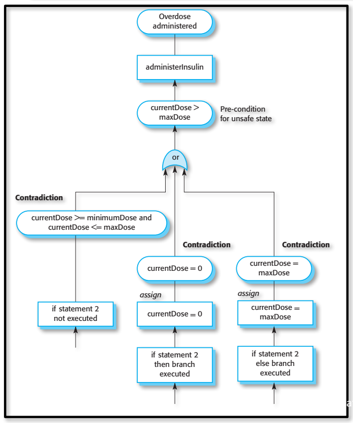
Insulin dose computation with safety checks

```
-- The insulin dose to be delivered is a function of blood sugar level,
-- the previous dose delivered and the time of delivery of the previous dose

currentDose = computeInsulin () ;

// Safety check—adjust currentDose if necessary.
// if statement 1
if (previousDose == 0)
{
            if (currentDose > maxDose/2)
            currentDose = maxDose/2 ;
}
else
            if (currentDose > (previousDose * 2) )
                        currentDose = previousDose * 2 ;
// if statement 2
if ( currentDose < minimumDose )
            currentDose = 0 ;
else if ( currentDose > maxDose )
            currentDose = maxDose ;
administerInsulin (currentDose) ;
```

Informal safety argument based on demonstrating contradictions



```
...

/* if statement 2 */
if ( currentDose < minimumDose ) {
    currentDose = 0 ;
}
else if ( currentDose > maxDose ) {
    currentDose = maxDose ;
}

administerInsulin (currentDose) ;
...
```

Program paths

```
if ( currentDose < minimumDose ) {
    currentDose = 0 ;
}
else if ( currentDose > maxDose ) {
    currentDose = maxDose ;
}
administerInsulin (currentDose) ;
```

- ✧ Neither branch of if-statement 2 is executed
  - Can only happen if CurrentDose is >= minimumDose and <= maxDose.
- ✧ then branch of if-statement 2 is executed
  - currentDose = 0.
- ✧ else branch of if-statement 2 is executed
  - currentDose = maxDose.
- ✧ In all cases, the post conditions contradict the unsafe condition that the dose administered is greater than maxDose.

Key points

- ✧ Safety-critical systems are systems whose failure can lead to human injury or death.
- ✧ A hazard-driven approach is used to understand the safety requirements for safety-critical systems. You identify potential hazards and decompose these (using methods such as fault tree analysis) to discover their root causes. You then specify requirements to avoid or recover from these problems.
- ✧ It is important to have a well-defined, certified process for safety-critical systems development. This should include the identification and monitoring of potential hazards.

Key points

- ✧ Static analysis is an approach to V & V that examines the source code of a system, looking for errors and anomalies. It allows all parts of a program to be checked, not just those parts that are exercised by system tests.
- ✧ Model checking is a formal approach to static analysis that exhaustively checks all states in a system for potential errors.
- ✧ Safety and dependability cases collect the evidence that demonstrates a system is safe and dependable. Safety cases are required when an external regulator must certify the system before it is used.