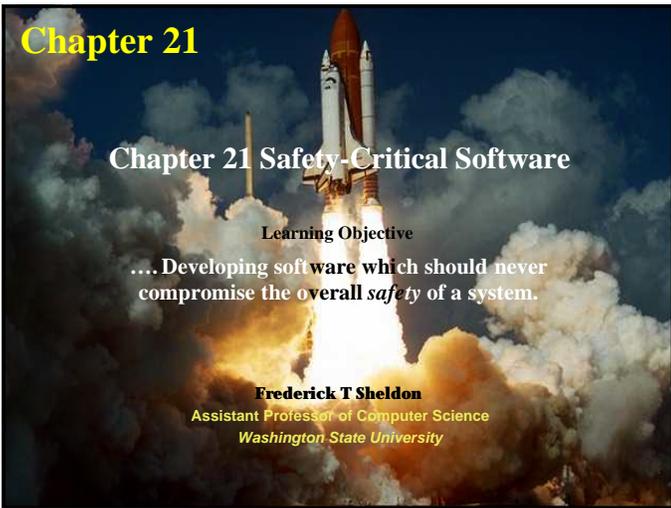


Chapter 21

Chapter 21 Safety-Critical Software

Learning Objective
.... Developing software which should never compromise the overall *safety* of a system.

Frederick T Sheldon
Assistant Professor of Computer Science
Washington State University



Objectives

- To introduce the concept of safety-critical software
- To describe the safety-critical system development process
- To introduce methods of process and product safety assurance

CS 442 Software Engineering Principles
From Software Engineering by L. Sommerville, 1996.

Chapter 21
Slide 2

Topics covered

- Definitions of safety-critical system terminology
- An insulin pump example
- Safety specification
- Hazard analysis
- Risk assessment and reduction
- Safety assurance
- Hazard logs
- Safety proofs

CS 442 Software Engineering Principles
From Software Engineering by L. Sommerville, 1996.

Chapter 21
Slide 3

Safety-critical systems

Systems whose failure can threaten human life or cause serious environmental damage

Increasingly important as computers replace simpler, hard-wired control systems

Hardware safety is often based on the physical properties of the hardware. Comparable techniques cannot be used with software

Safety criticality

Primary safety-critical systems

Embedded software systems whose failure can cause the associated hardware to fail and directly threaten people.

Secondary safety-critical systems

Systems whose failure results in faults in other systems which can threaten people

Discussion here focuses on primary safety-critical systems

Secondary safety-critical systems can only be considered on a one-off basis

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

Damage

A measure of the loss resulting from a mishap

Hazard

A condition with potential for causing or contributing to a mishap

Hazard severity

An assessment of the worst possible damage which could result from a particular hazard

Definitions

Hazard probability

The probability of the events occurring which create a hazard

Risk

This is a complex concept which is related to the hazard severity, the hazard probability and the probability that the hazard will result in a mishap.

It is a measure of the probability that the system will behave in a way which threatens humans. The objective of all safety systems is to minimize risk.

Safety achievement

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 achievement should ensure that either these faults cannot occur or, if they do occur, they cannot result in a mishap

Should also ensure that correct functioning of the system does not cause a mishap

Safety and reliability

Not the same thing

Reliability is concerned with conformance to a given specification and delivery of service

Safety is concerned with ensuring system cannot cause damage irrespective of whether or not it conforms to its specification

Unsafe reliable systems

Specification errors

If the system specification is incorrect then the system can behave as specified but still cause an accident

Hardware failures generating spurious inputs

Hard to anticipate in the specification

Context-sensitive commands i.e. issuing the right command at the wrong time

Often the result of operator error

Accident occurrence

System design should always be based around the notion that no single point of failure can compromise system safety

However, accidents usually arise because of several simultaneous failures rather than a failure of a single part of the system

Software control

Adds complexity so hence may decrease overall system safety

BUT also allows a larger number of system parameters to be monitored, allows the use of inherently reliable electronic equipment and can be used to provide sophisticated safety interlocks

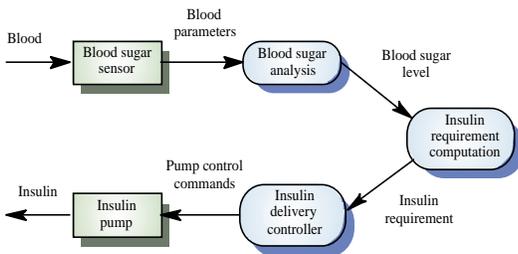
Therefore, software control may improve overall system safety even when occasional software failures occur

Insulin delivery

Simple example of a safety-critical system. Most medical systems are safety-critical
People with diabetes cannot make their own insulin (used to metabolize sugar). It must be delivered externally
Delivers a dose of insulin (required by diabetics) depending on the value of a blood sugar sensor

Insulin delivery system

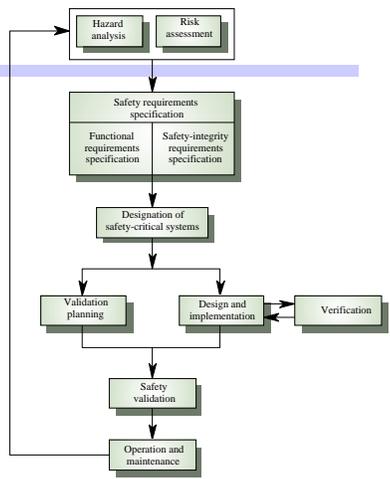
Data flow model of software-controlled insulin pump



Safety specification

The safety requirements of a system should be separately specified
These requirements should be based on an analysis of the possible hazards and risks
Safety requirements usually apply to the system as a whole rather than to individual sub-systems

The safety life-cycle



Safety processes

Hazard and risk analysis

Assess the hazards and the risks of damage associated with the system

Safety requirements specification

Specify a set of safety requirements which apply to the system

Designation of safety-critical systems

Identify the sub-systems whose incorrect operation may compromise system safety

Safety validation

Check the overall system safety

Hazard analysis

Identification of hazards which can arise

Structured into various classes of hazard analysis and carried out throughout software process

A risk analysis should be carried out and documented for each identified hazard

Hazard analysis stages

Hazard identification

Identify potential hazards which may arise

Hazard classification

Assess the risk associated with each hazard

Hazard decomposition

Decompose hazards to discover their potential root causes

Safety specification

Define how each hazard must be taken into account when the system is designed

Structured hazard analysis

For large systems, hazard analysis must be structured

Preliminary hazard analysis Assess the principal hazards for the system in its operating environment

Sub-system hazard analysis Assess hazards for each safety-critical sub-system

System hazard analysis Assess hazards which result from sub-system interaction

Software hazard analysis Assess hazards related to incorrect software function

Operational hazard analysis Assess hazards resulting from incorrect system use

Insulin system hazards

insulin overdose or -

power failure

machine interferes electrically with other medical equipment such as a heart pacemaker

parts of machine break off in patient's body

poor sensor/actuator contact

infection caused by introduction of machine

allergic reaction to the materials or insulin used in the machine

Fault-tree analysis

Method of hazard analysis which starts with an identified fault and works backward to the causes of the fault.

Can be used at all stages of hazard analysis from preliminary analysis through to detailed software checking

Top-down hazard analysis method. May be combined with bottom-up methods which start with system failures and lead to hazards

Fault- tree analysis

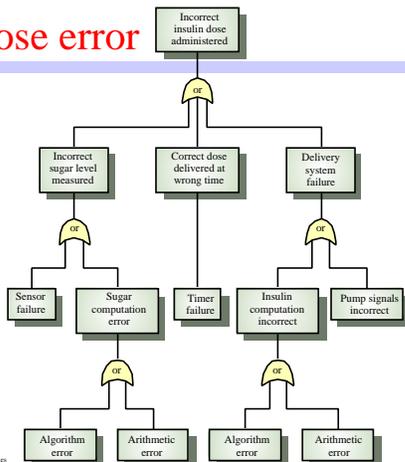
Identify hazard

Identify potential causes of the hazard. Usually there will be a number of alternative causes. Link these on the fault-tree with 'or' or 'and' symbols

Continue process until root causes are identified

A design objective should be that no single cause can result in a hazard. That is, 'or's should be replaced by 'and's wherever possible

Insulin dose error



Risk assessment

Assesses hazard severity, hazard probability and accident probability

Outcome of risk assessment is a statement of acceptability

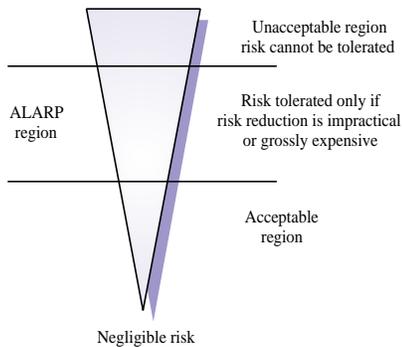
Intolerable. Must never arise or result in an accident

As low as reasonably practical(ALARP) Must minimize possibility of hazard given cost and schedule constraints

Acceptable. Consequences of hazard are acceptable and no extra costs should be incurred to reduce hazard probability

Levels of risk

Width of triangle is proportional to the cost of dealing with the hazard



Risk acceptability

The acceptability of a risk is determined by human, social and political considerations

In most societies, the boundaries between the regions are pushed upwards with time i.e. society is less willing to accept risk

For example, the costs of cleaning up pollution may be less than the costs of preventing it but this may not be socially acceptable

Risk assessment is subjective

Risks are identified as probable, unlikely, etc. This depends on who is making the assessment

Risk analysis example

Identified hazard	Hazard probability	Hazard severity	Estimated risk	Acceptability
Insulin overdose	Medium	High	High	Intolerable
Insulin underdose	Medium	Low	Low	Acceptable
Power failure	High	Low	Low	Acceptable
Machine incorrectly fitted	High	High	High	Intolerable
Machine breaks in patient	Low	High	Medium	ALARP
Machine causes infection	Medium	Medium	Medium	ALARP
Electrical interference	Low	High	Medium	ALARP
Allergic reaction	Low	Low	Low	Acceptable

Risk reduction

System should be specified so that hazards do not arise or result in an accident

Hazard avoidance

The system should be designed so that the hazard can never arise during correct system operation

Hazard probability reduction

The system should be designed so that the probability of a hazard arising is minimized

Accident prevention

If the hazard arises, there should be mechanisms built into the system to prevent an accident

Insulin delivery system

Safe state is a shutdown state where no insulin is delivered

If hazard arises, shutting down the system will prevent an accident

Software may be included to detect and prevent hazards such as power failure

Consider only hazards arising from software failure

Arithmetic error: insulin dose is computed incorrectly because of some failure of the computer arithmetic

Algorithmic error: dose computation algorithm is incorrect

Arithmetic errors

Use language exception handling mechanisms to trap errors as they arise

Use explicit error checks for all errors which are identified

Avoid error-prone arithmetic operations (multiply and divide). Replace with add and subtract

Never use floating-point numbers

Shut down system if error detected (safe state)

Algorithmic errors

Harder to detect. System should always err on the side of safety

Use reasonableness checks for the dose delivered based on previous dose and rate of dose change

Set maximum delivery level in any specified time period

If computed dose is very high, medical intervention may be necessary anyway because the patient may be ill

Safety assurance

Avoid safety problems by using 'safe' design techniques

Ensure that the software process has appropriate safety reviews and checks

Apply explicit safety assurance techniques to the developed software

Design principles for safe software

Make software as simple as possible

Use simple techniques for software development avoiding error-prone constructs such as pointers and recursion

Use information hiding to localize the effect of any data corruption

Make appropriate use of fault-tolerant techniques but do not be seduced into thinking that fault-tolerant software is necessarily safe

Formal methods and safety

Formal methods are mandated in Britain for the development of some types of safety-critical software

Formal specification and proof increases confidence that a system meets its specification

Formal specifications require specialized notations so domain experts cannot check for specification incompleteness

The cost-effectiveness of formal methods is unknown

Use of formal methods for safety-critical software development is likely to increase

Process assurance

The software process should be designed to include the collection of safety-related information and should include safety reviews

- Hazard logging and monitoring

- Explicit identification of project safety engineers

- Safety reviews

- Safety certification

- Detailed configuration management to ensure that the delivered system is the one which has been checked for safety

The hazard log tracks the documentation and management of hazards

Hazard log entry

Hazard Log. Page 4: Printed 21.

System: *Insulin Delivery System* File: *Insulin System/Safety/HLog*
Safety Engineer: *James Brown* Log version: *1.3*

Identified Hazard: *Insulin overdose delivered to patient*

Identified by: *Jane Williams*

Criticality Class: *1*

Identified Risk: *Moderate*

Fault tree identified: *YES* Date: *10.11.90* Location: *Hazard Log, Page 5*

Fault tree creator: *Jane Williams and Bill Smith*

Fault tree checked: *YES* Date: *20.11.90* Checker: *James Brown*

System design safety requirements:

1. Incorporate self-testing software for sensor system, clock and delivery system. This should be executed at least once per minute and should cause an audible warning to be emitted if a fault is discovered. If a fault is discovered, no further insulin deliveries should be made until the system has been reset.
2. Incorporate a patient override facility so that the patient may modify the dose to be delivered by manual intervention. However, a limit should be set on the dose administered by the patient. This limit should be set by medical staff when the system is installed.
3. ...

Safety reviews

- Review for correct intended function
- Review for maintainable, understandable structure
- Review to verify algorithm and data structure design against specification
- Review to check code consistency with algorithm and data structure design
- Review adequacy of system testing

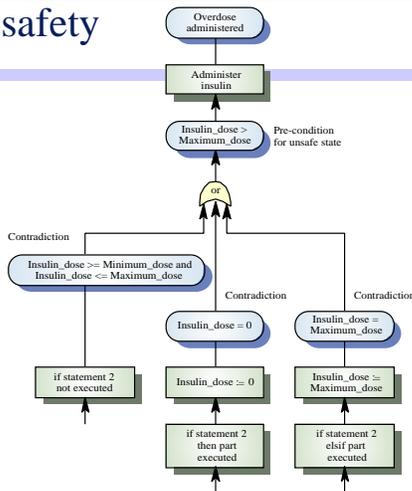
Safety proofs

- Safety proofs are intended to show that the system cannot reach in unsafe state
- Weaker than correctness proofs which must show that the system code conforms to its specification
- Generally based on proof by contradiction
 - Assume that an unsafe state can be reached
 - Show that this is contradicted by the program code
- May be displayed graphically

Insulin delivery code

```
-- 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
Insulin_dose := Compute_insulin ( Blood_sugar_level,
    Previous_dose, Previous_time );
-- if statement 1
if Insulin_dose > Previous_dose + Previous_dose then
    Insulin_dose := Previous_dose + Previous_dose ;
end if ;
-- Don't administer very small doses
-- if statement 2
if Insulin_dose < Minimum_dose then
    Insulin_dose := 0 ;
-- Don't deliver more than maximum dose
elseif Insulin_dose > Maximum_dose then
    Insulin_dose := Maximum_dose ;
end if ;
-- root of fault tree
-- if statement 3
if Insulin_dose > 0 then
    Administer_insulin (Insulin_dose) ;
end if ;
```

Informal safety proof



Safety assertions

Predicates included in the program indicating conditions which should hold at that point
May be based on pre-computed limits e.g. number of insulin pump increments in maximum dose
Used in formal program inspections or may be pre-processed into safety checks

Safety assertions

```
procedure Administer_insulin (Insulin_dose: DOSE) is  
  Insulin_increments: NATURAL ;  
begin  
  --* assert Insulin_dose <= Maximum_dose  
  Insulin_increments := Compute_requirement (Insulin_dose) ;  
  --* assert Insulin_increments <= Maximum_increments  
  for i in range 1..Insulin_increments loop  
    Generate_pump_signal ;  
    --* assert i <= Maximum_increments ;  
  end loop ;  
end Administer_insulin ;
```

Key points

Safety is a system property
Hardware safety methods are not completely applicable to safety-critical software
Software control can improve safety by providing more checking and interlocks
The development process for safety-critical software is important
Hazard analysis is a key part of the safety specification process

Key points

Risk analysis involves assessing the probability of hazards, their severity and the probability that they will result in an accident
Design strategies may be used for hazard avoidance, hazard probability reduction and accident avoidance
Safety assurance depends on professional skills
Safety proofs may be used as part of product safety assurance
