AARHUS
UNIVERSITY
DEPARTMENT OF ENGINEERING

SOFTWARE ENGINEERING PRINCIPLES REQUIREMENTS FOR CRITICAL SYSTEMS

STEFAN HALLESTEDE
PETER GORM LARSEN
CARL SCHULTZ

CRITICAL SYSTEMS CATEGORIES

- Safety Critical Systems: Failure leads to loss of
  - Life
  - Health
  - Well-being
- Mission Critical Systems: Failure leads to loss of
  - Business
  - Equipment
  - Effort
- Security Critical Systems: Failure leads to loss of
  - Confidentiality
  - Integrity
  - Availability of service

SAFETY CRITICAL SYSTEMS

- **Safety**: a property of a system that it will not endanger human life or the environment

- **Safety-critical system**: a system by which the safety of equipment or plant is assured

FAILSAFE SYSTEMS

› **Failsafe system**:

› **Failure**:

› **Error**:

› **Fault**:

FAILSAFE SYSTEMS

- **Failsafe system**: a system designed to fail in a safe state, e.g., trains can be made to stop in the case of signal **failure**
- **Failure**: the event of a system or component failing to perform, or deviating from its intended/required function for a specified time under specified environment conditions
- **Error**: deviation from an intended/required, correct state of the system or subsystem
- **Fault**: the real or hypothesised cause of an error (actual or potential)

FAILURES AND FAULTS

- All failures are faults, **but** not all faults are failures
- Example: A relay closing when it should not is a fault.
- If the **fault** was caused by a problem within the relay, it is also **a failure** of the relay.
- If the **fault** was caused by a spurious signal from some other component, it is **not a failure** of the relay.

PRIMARY AND SECONDARY FAULTS

- **Primary fault**:

- **Secondary fault**:

PRIMARY AND SECONDARY FAULTS

- **Primary fault**: a system or component fails **within** the intended operating environment or conditions
- **Secondary fault**: a system or component fails **outside** its intended operating environment or conditions

USEFUL LIFETIME

- **Early failures**: occur during a **debugging** or **burn-in period**
- **Random failures**: result from complex, uncontrollable causes usually primarily considered **during the useful life** of the component or system;
  (*Difficult to apply to software! Why?*)
- **Wearout failures**: occur when the components or systems are **past their useful life**

SAFETY-CRITICAL SYSTEM CASE: THERAC-25

- History of accidents involving the radiation therapy machine Therac-25

- This case presentation follows
- N. G. Leveson, Medical Devices: The Therac-25, 1995 http://sunnyday.mit.edu/papers/therac.pdf
- N. G. Leveson, Safeware: System Safety and Computers, Addison-Wesley, 1995, Appendix A

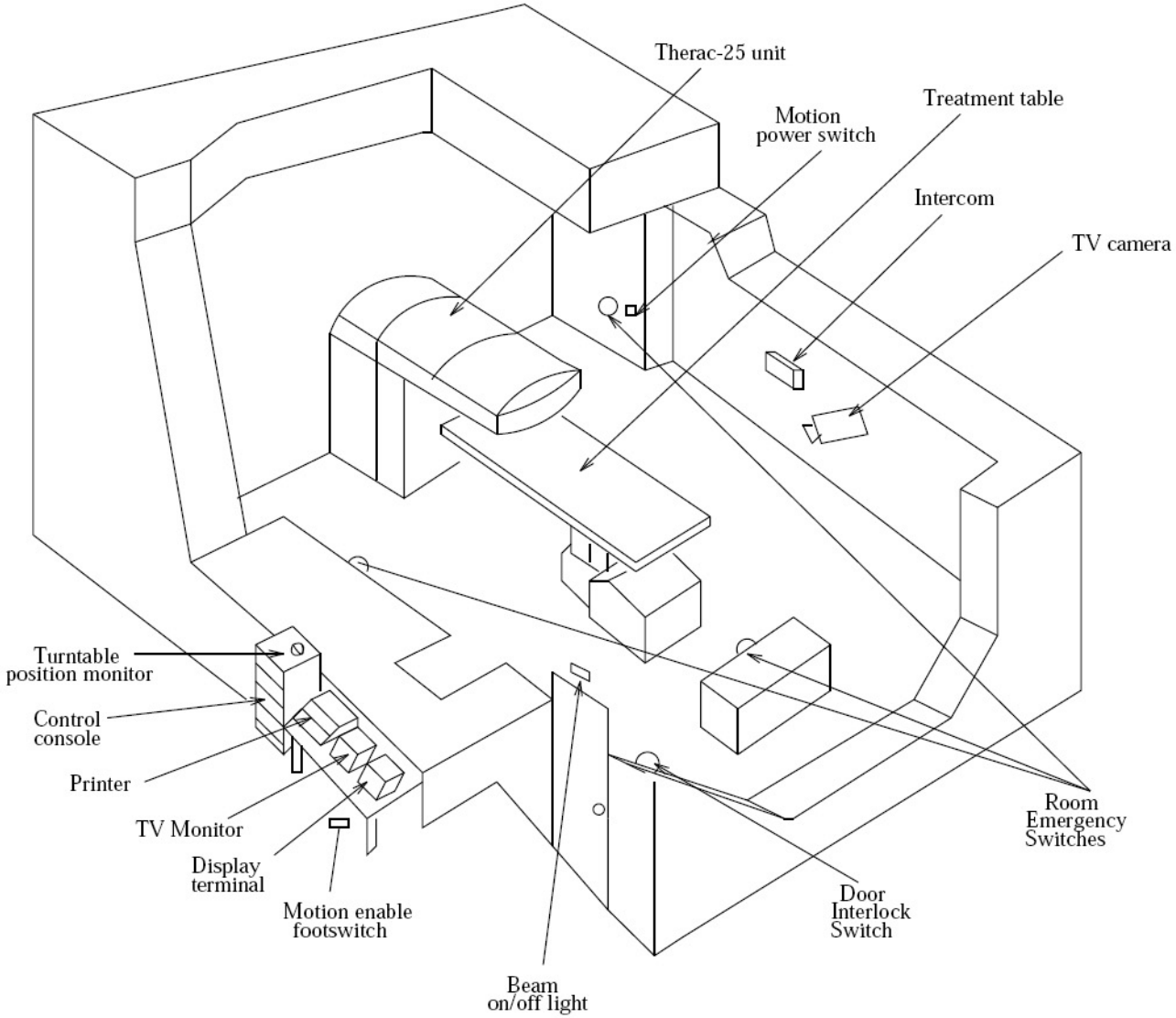
THERAC-25 CASE SUMMARY

- Therac-25 is a **computer-controlled radiation therapy machine** used in medicine.
- Built by a Canadian company, AECL. Installed inseveral hospitals in Canada and the US.
- Between June 1985 and July 1987, 6 people wrongly received **massive overdoses of radiation** — 4 of these **died** as a result.
- Dual mode device:
  - Electron beam - used to treat shallow tissue.
  - Photon Beam - used to treat deep tissue.
- Also, alignment mode using mirror & (normal) lightbeam for correct positioning of radiation beam. Each mode requires a **different position** of the turntable.

HISTORY OF THE DEVICE

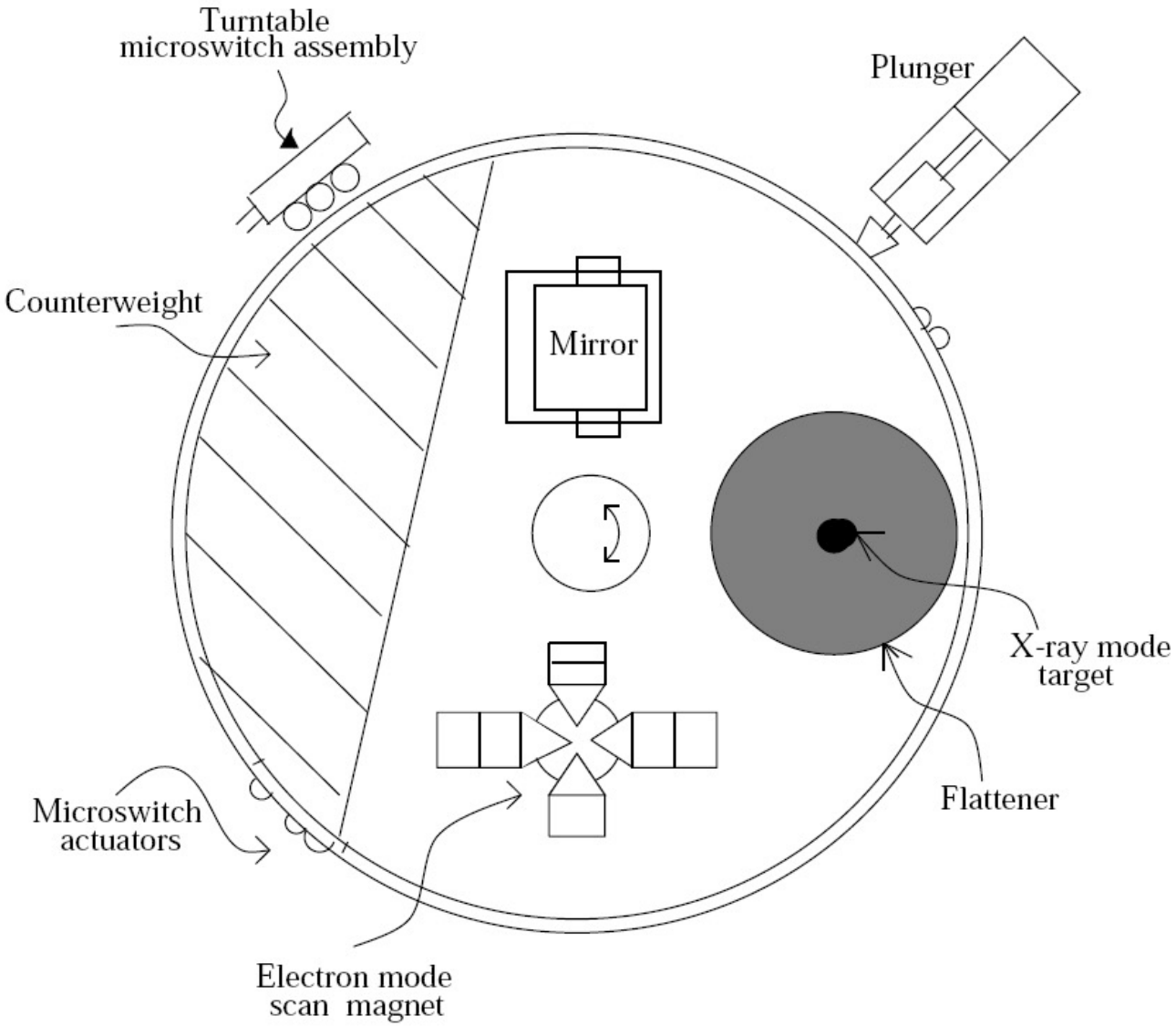
- Therac-6: Photon beam only device. Linked to a DEC PDP-11 for operator convenience, but capable of being operated without the computer.
  Safety implemented in hardware.
- Therac-20: Dual mode device. Again, linked to a DEC PDP-11 for operator convenience, but capable of being operated without the computer.
  Safety implemented in hardware.
- Therac-25: Dual mode device. PDP-11 required for operation.
  Relies on software for some safety features.
  All software written in assembly language.
  Therac-25 reused software from Therac-20 and Therac-6.

A TYPICAL THERAC-25 FACILITY



Source: Leveson, Medical Devices: The Therac-25, 1995

THE TURNTABLE



Source: Leveson, Medical Devices: The Therac-25, 1995

THE CONSOLE

PATIENT NAME : TEST
TREATMENT MODE : FIX BEAM TYPE: X ENERGY (MeV): 25

	ACTUAL	PRESCRIBED	
UNIT RATE/MINUTE	0	200	
MONITOR UNITS	50 50	200	
TIME (MIN)	0.27	1.00	
GANTRY ROTATION (DEG)	0.0	0	VERIFIED
COLLIMATOR ROTATION (DEG)	359.2	359	VERIFIED
COLLIMATOR X (CM)	14.2	14.3	VERIFIED
COLLIMATOR Y (CM)	27.2	27.3	VERIFIED
WEDGE NUMBER	1	1	VERIFIED
ACCESSORY NUMBER	0	0	VERIFIED

DATE : 84-OCT-26 SYSTEM : BEAM READY OP. MODE : TREAT AUTO
TIME : 12:55: 8 TREAT : TREAT PAUSE X-RAY 173777
OPR ID : T25V02-R03 REASON : OPERATOR COMMAND:

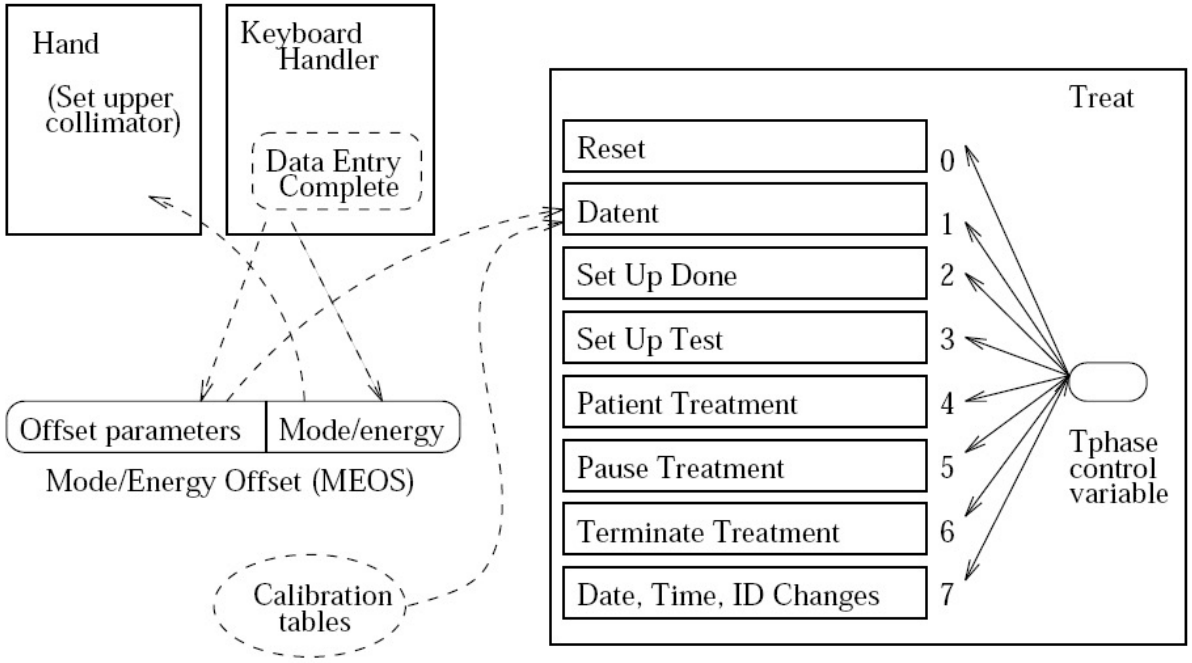
Source: Leveson, Medical Devices: The Therac-25, 1995

FIRST FAULT (MAY 1986)

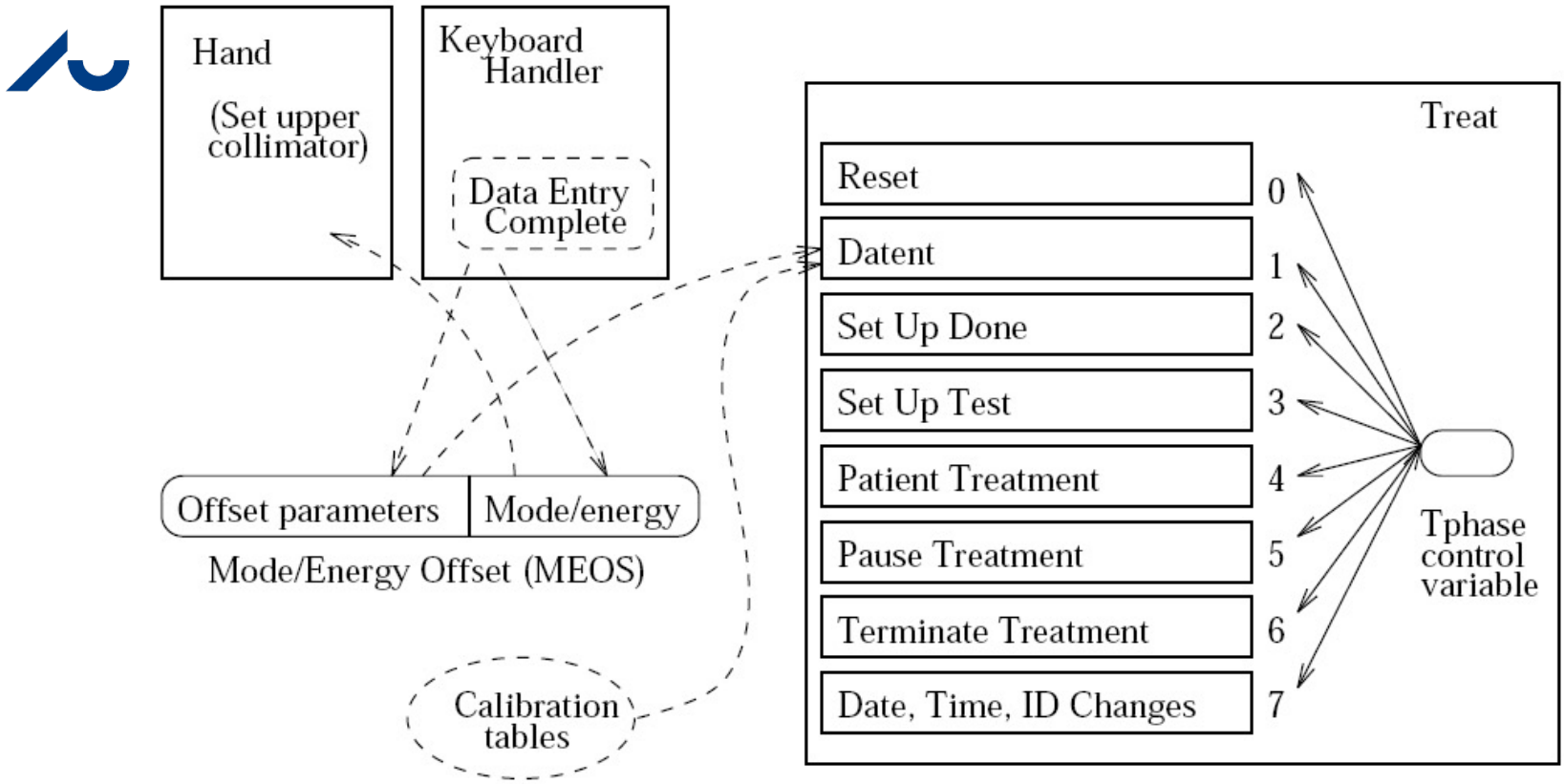
- *Changes to parameters can be made during setup. Setup takes 8 seconds.*
- **Problem**: in some cases, these changes are ignored by the system even though they are reflected on the screen. Caused by a flag being reset in the wrong place.
- **Accident scenario**: Operator selected photon by mistake, setup was initiated, then operator changed mode and energy level within 8 seconds. These changes were ignored, so when the beam was activated, the dosage was much too high.
- "**Fix**": remove the ↑ key from the keyboard to prevent edits during setup. Device now has to be reset to change parameters.

HAS THIS PROBLEM BEEN SOLVED?

› Unsynchronised shared memory access
› No consistency checks of set parameters



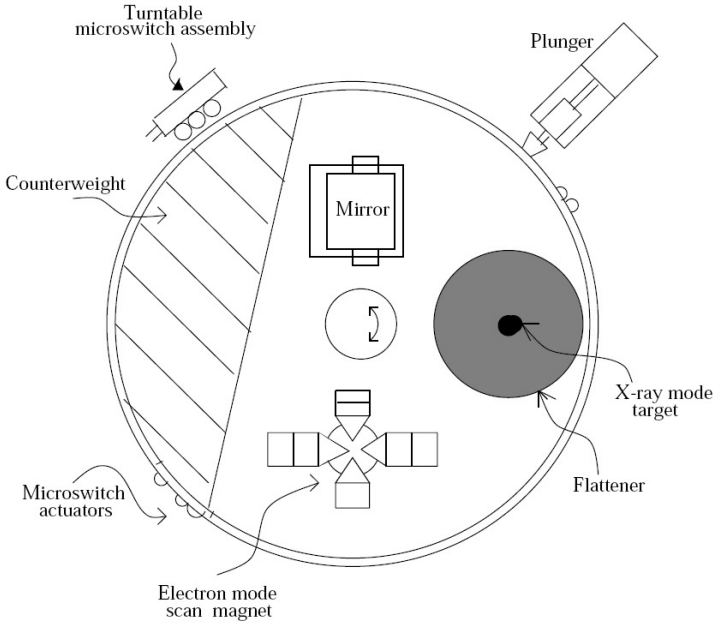
› Source: Leveson, Medical Devices: The Therac-25, 1995



PATIENT NAME : TEST
TREATMENT MODE : FIX BEAM TYPE: X ENERGY (MeV): 25

	ACTUAL	PRESCRIBED	
UNIT RATE/MINUTE	0	200	
MONITOR UNITS	50 50	200	
TIME (MIN)	0.27	1.00	
GANTRY ROTATION (DEG)	0.0	0	VERIFIED
COLLIMATOR ROTATION (DEG)	359.2	359	VERIFIED
COLLIMATOR X (CM)	14.2	14.3	VERIFIED
COLLIMATOR Y (CM)	27.2	27.3	VERIFIED
WEDGE NUMBER	1	1	VERIFIED
ACCESSORY NUMBER	0	0	VERIFIED

DATE : 84-OCT-26 SYSTEM : BEAM READY OP. MODE : TREAT AUTO
TIME : 12:55: 8 TREAT : TREAT PAUSE X-RAY 173777
OPR ID : T25V02-R03 REASON : OPERATOR COMMAND:



THE “FIX” (LETTER TO CUSTOMERS):

SUBJECT: CHANGE IN OPERATING PROCEDURES FOR THE THERAC 25 LINEAR ACCELERATOR

Effective immediately, and until further notice, the key used for moving the cursor back through the prescription sequence (i.e., cursor "UP" inscribed with an upward pointing arrow) must not be used for editing or any other purpose. To avoid accidental use of this key, the key cap must be removed and the switch contacts fixed in the open position with electrical tape or other insulating material. For assistance with the latter you should contact your local AECL service representative.

Disabling this key means that if any prescription data entered is incorrect then [an] "R" reset command must be used and the whole prescription reentered. For those users of the Multiport option, it also means that editing of dose rate, dose, and time will not be possible between ports.

THE AUTHORITIES (FDA) RESPONSE (LETTER TO AECL):

We have reviewed Mr. Downs' April 15 letter to purchasers and have concluded that it does not satisfy the requirements for notification to purchasers of a defect in an electronic product. Specifically, it does not describe the defect nor the hazards associated with it. The letter does not provide any reason for disabling the cursor key and the tone is not commensurate with the urgency for doing so. In fact, the letter implies the inconvenience to operators outweighs the need to disable the key. We request that you immediately renotify purchasers.

SECOND FAULT (JAN 1987)

- **Problem**: There is a software interlock which prevents activation of the beam when the turntable is not correctly positioned. This failed, allowing beam to be activated when turntable was in the incorrect position.
- **Cause**: A flag was set using the code:

```
flag := flag+1
```

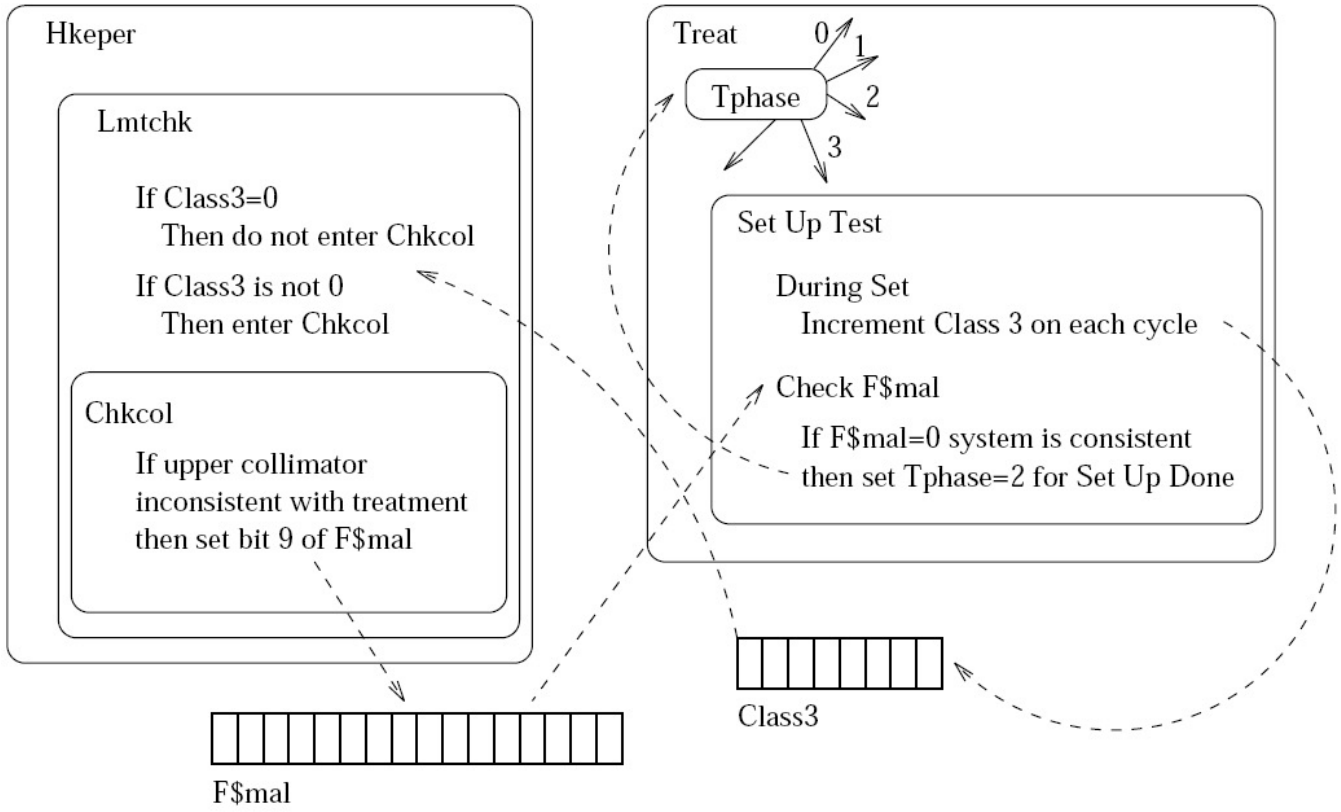
SECOND FAULT (JAN 1987)

- **Problem**: There is a software interlock which prevents activation of the beam when the turntable is not correctly positioned. This failed, allowing beam to be activated when turntable was in the incorrect position.
- **Cause**: A flag was set using the code:

```
flag := flag+1
```

  This was an 8-bit integer so it wrapped around to zero!
- "**Fix**":Software fixed. Extra hardware interlocks and shutdown systems added.

ANALYSIS OF SECOND FAULT



Source: Leveson, Medical Devices: The Therac-25, 1995

THE WORKING MACHINE

› Would you trust this machine
now
that the faults have been fixed?

› Why?

LESSONS LEARNED (LEVESON)

Process:

› Overconfidence in software.
› Confusing reliability with safety.
› Lack of defensive design.
› Failure to eliminate root causes.
› Complacency.
› Unrealistic risk assessments.
› Inadequate investigation or follow-up on accident reports.
› Dangerous reuse of software.
› Safe versus friendly user interface
› Importance of government regulator (FDA).

LESSONS LEARNED (LEVESON)

Inadequate software engineering practices:

- Software specification and documentation were an afterthought.
- Bad quality assurance procedures.
- Software design was unnecessarily complex.
- No audit trails.
- Inadequate testing and formal analysis of code.
- Bad user interface design, in particular, incomprehensible error messages

SAFETY IN SYSTEM DEVELOPMENT

- Safety should be treated as a **separate** (though related) issue to other system requirements during development
- Important to have a **safety plan**

- Safety plan :
  - set out the manner in which **safety** will be **achieved**
  - define management structure responsible for **hazard** and **risk** analysis
  - identify key **staff**
  - assign **responsibilities** within projects performed by large companies or consortia

SOFTWARE IN CRITICAL SYSTEMS

- ***Advantages***
  - Allows more **complex processes** to be controlled
  - Added **flexibility**

- ***Disadvantages***
  - More **complexity** increases the scope for **errors**
  - **Flexibility** makes it easy to introduce **errors**
  - **Inherent complexity** of software
  - Exhaustive testing **impossible**
  - Potential to provide **too much information** leading to confusion

(SOFTWARE) STANDARDS

- Help to ensure that a product meets certain levels of **quality**
- Help to establish that a product has been developed using methods of **known effectiveness**
- Promote **uniformity of approach** between different teams
- Provide **guidance** on design and development techniques
- Provide **legal basis** in case of dispute

PROGRAMMING LANGUAGES FOR CRITICAL SYSTEMS

- Desirable language features include:
  - Logical soundness and a clear precise definition
  - Expressive power
  - Strong typing
  - Verifiability
  - Security (language violations can be checked statically)
  - Bounded space requirements
  - Bounded time requirements
  - Reliable compilers
  - Good tool support

THE CHOICE OF LANGUAGE

```
#include <stdio.h>

int main(void) {
    printf("Hello, world!\n");
    return 0;
}
```

”Hello World”
in C
and in
C-INTERCAL

```
DO ,1 <- #13
PLEASE DO ,1 SUB #1 <- #238
DO ,1 SUB #2 <- #108
DO ,1 SUB #3 <- #112
DO ,1 SUB #4 <- #0
DO ,1 SUB #5 <- #64
DO ,1 SUB #6 <- #194
DO ,1 SUB #7 <- #48
PLEASE DO ,1 SUB #8 <- #22
DO ,1 SUB #9 <- #248
DO ,1 SUB #10 <- #168
DO ,1 SUB #11 <- #24
DO ,1 SUB #12 <- #16
DO ,1 SUB #13 <- #162
PLEASE READ OUT ,1
PLEASE GIVE UP
```

COMMON SOFTWARE FAULTS

- Subprogram side-effects
- Aliasing
- Failure to initialise
- Expression evaluation errors
- Control flow errors

- Some of these can be avoided by
- *Restricting constructs of programming languages*

LANGUAGE SUBSETS

- Use of a subset of a programming language with some or all of these features.

- Spark Ada:
  - No gotos
  - No pointers
  - No dynamic data structures
  - No exceptions
  - No tasking
  - No side-effects
  - No Generics

- What about Java?

SOFTWARE ANNOTATIONS

- Information embedded in program code that is used by **static analysis tools** but *ignored by a compiler*.
- Examples of annotations include
  - Lists of the global variables used by subprograms and dependencies between variables. These are used for checking coding errors such as **using variables before initialising** them.
  - Preconditions and postconditions which are used for the generation of proof obligations in formal verification, e.g., to **avoid passing null-pointers**.

STATIC ANALYSIS

- Investigate properties of a system **without operating it**:
  - Type checking
  - Formal verification / refinement
  - Control flow analysis (e.g., Spark)
  - Walkthroughs / design reviews
  - Metrics – algorithmic complexity, module complexity measures (Cyclomatic/McCabe etc)

SPARK ADA EXAMPLE OF ANNOTATIONS

```
procedure Exchange(X, Y: in out Float)
--# derives X from Y &
--#         Y from X ;
--# post X=Y and Y=X ;
  is
    T : Float;
  begin
    T:=X; X:=Y; Y:=X;
  end Exchange;
```

SPARK ADA EXAMPLE OF ANNOTATIONS

```
procedure Exchange(X, Y: in out Float)
--# derives X from Y &
--#             Y from X ;
--# post X=Y and Y=X ;
   is
      T : Float;
   begin
      T:=X; X:=Y; Y:=X;
   end Exchange;
```

Spark Examiner Output:

```
!!! (1)   Ineffective statement: T := X.
!!! (2)   Importation of initial value of X
          ineffective.
!!! (3)   Variable T is neither
          referenced not exported.
!!! (4)   Imported value of X not used
          in derivation of Y.
??? (4)   Imported value of Y may be
          used in derivation of Y.
```

LOGICAL ASSERTIONS

```
…
assert sorted(a);
do_something_with(a);
assert descending(a);
…
```

What if sorted contains assertions?

LOGICAL ASSERTIONS

```
…
int ass1 = sorted(a);
assert ass1;
do_something_with(a);
int ass2 = descending(a)
assert ass2;
…
```

LOGICAL ASSERTIONS

```
...
assert x = (y /= z);
if (x) y = z;
assert y == z;
...
```

What does this program do?

LOGICAL ASSERTIONS

```
int x[1000000];
int y[1000000];
...
for (i=0; i<10000000; i++)
  assert x[i] < y[i];
...
```

What does this program do?

LOGICAL ASSERTIONS

- Only use “simple” formulas (no function calls)
- No side effects!
- Make sure expressions in assertions are defined
- Avoid assertions that need a long time to evaluate

- Later in the course: **prove truth of assertions**

DYNAMIC ANALYSIS (TESTING)

- Functional Testing (**Black Box**):
  - Use test cases to check that the system has the required functionality.
  - No knowledge of implementation assumed.
  - Test cases generated from (formal) specification.
- Structural Testing (**White Box**):
  - Investigate characteristics of systems using detailed knowledge of implementation.
  - Devise test cases to check individual subprograms / execution paths.
- Random Testing:
  - Random selection of test cases.
  - Aims to detect faults that may be missed by more systematic techniques.
- Test Coverage:
  - Statement, branch(decision), MCDC, call graph, ...

SPECIFIC TESTING TECHNIQUES

- Equivalence partitioning of test cases
- Boundary value analysis (test software at, and at either side of, each boundary value)
- Error guessing of possible faults based on experience
- Error seeding: deliberate planting of errors to determine effectiveness of testing
- Performance testing
- Stress testing

ENVIRONMENTAL SIMULATION

- For control/protection systems, it is impossible to fully test a system within its operational environment
- Simulator of the environment used instead
- Often more complex than system under test!
- Issues:
  - Which environmental variables are included
  - Accuracy of models
  - Accuracy of calculations
  - Timing considerations

MAINTENANCE

- **Maintenance**: the action taken to retain a system in, or return a system to, its design operation condition

- **Maintainability**: the ability of a system to be maintained

SAFETY

- **Accident**:
  - unintended event or sequence of events that causes death, injury, environmental or material damage
  - unplanned event that results in a certain level of damage or loss to human life or the environment
- **Incident** :
  - unplanned event that involves no loss or damage, but has the potential to be an accident in different circumstance
- **Safety**: freedom of a system from accidents or losses

EXAMPLE OF ACCIDENT AND INCIDENT

- **Accident**: two planes crash

- **Incident**: two planes get very close leaving no margin for manoeuvres ("near miss")

RELIABILITY

- **Reliability**: the probability that a system or component will perform its intended function satisfactorily for a prescribed period of time under a given set of operating conditions

- **Safety and reliability**: a system can be unreliable but safe if it does not behave according to its specification but still does not cause an accident

HAZARD

- **Hazard**: a situation in which there is the danger of an accident occurring
- **Hazard severity**: the worst possible accident that could result from the hazard
- **Hazard likelihood/probability**: can be specified qualitatively or quantitatively when a new system is designed usually no historical data is available so qualitative evaluation may be the best that can be done

EXAMPLE: HAZARD SEVERITY CATEGORIES FOR CIVIL AIRCRAFT

Category	***Definition***
Catastrophic	Failure conditions that would prevent continued safe flight and landing
Hazardous	Failure conditions that would reduce the capability of the aircraft or the ability of the crew to cope with adverse operating conditions; per- haps small number of occupants injured
Major	Failure conditions that would reduce the capability of the aircraft or the ability of the crew to cope with adverse operating conditions; causing discomfort to occupants
Minor	Failure conditions that would not significantly reduce aircraft safety
No effect	Failure conditions that would not affect the operational capability of the aircraft or increase crew workload

RISK

- **Risk**: combination of the **severity** of a specified hazardous event with its **probability** of occurrence over a specified duration
- **Example**
- Failure of a particular component results in an explosion that could kill 100 people. It is estimated that the component will fail once every 10000 years.
- *What is the risk associated with this component?*
- Risk = severity × probability of occurrence per year
  = 100 × (1/10000)
  = 0.01 deaths per year

RISK ASSESSMENT EXERCISE

- In a country with a population of 50.000.000, approximately 25 people are killed each year by lightning

What is the risk associated with death from this cause?

- *The fraction of the population killed per year is 25/50.000.000, i.e, 5 × 10−7*
- *Each individual has a probability of 5×10−7 of being killed in a given year*
- *risk = 5 × 10−7 deaths per person year*

What is a specific persons risk of getting killed by lightnining?

HAZARD SEVERITY CATEGORIES (IEC 1508)

Category	Definition
Catastrophic	Multiple deaths
Critical	A single death, and/or multiple severe injuries or severe occupational illnesses
Marginal	A single severe injury or occupational illness, and/or multiple minor injuries or minor occupational illnesses
Negligible	At most a single minor injury or minor occupational illness

HAZARD PROBABILITY RANGES (IEC 1508)

Frequency	Occurrences during operational life	Probability
Frequent	Likely to be continually experienced	$[10^{0} – 10^{-2})$
Probable	Likely to occur often	$[10^{-2} – 10^{-4})$
Occasional	Likely to occur several times	$[10^{4} – 10^{-6})$
Remote	Likely to occur some time	$[10^{-6} – 10^{--8})$
Improbable	Unlikely, but may exceptionally occur	$[10^{-8} – 10^{-9})$
Incredible	Extremely unlikely to occur at all	$[10^{-9} – ...)$

NB: The numbers are invented and are not part of the standard

RISK CLASSES (IEC 1508)

Risk class	Interpretation
I	Intolerable risk
II	Undesirable risk, and tolerable only if risk reduction is impracticable or if costs are grossly disproportionate to the improvement gained
III	Tolerable risk if the cost of risk reduction would exceed the improvement gained
IV	Negligible risk

RISK CLASSIFICATION (IEC 1508)

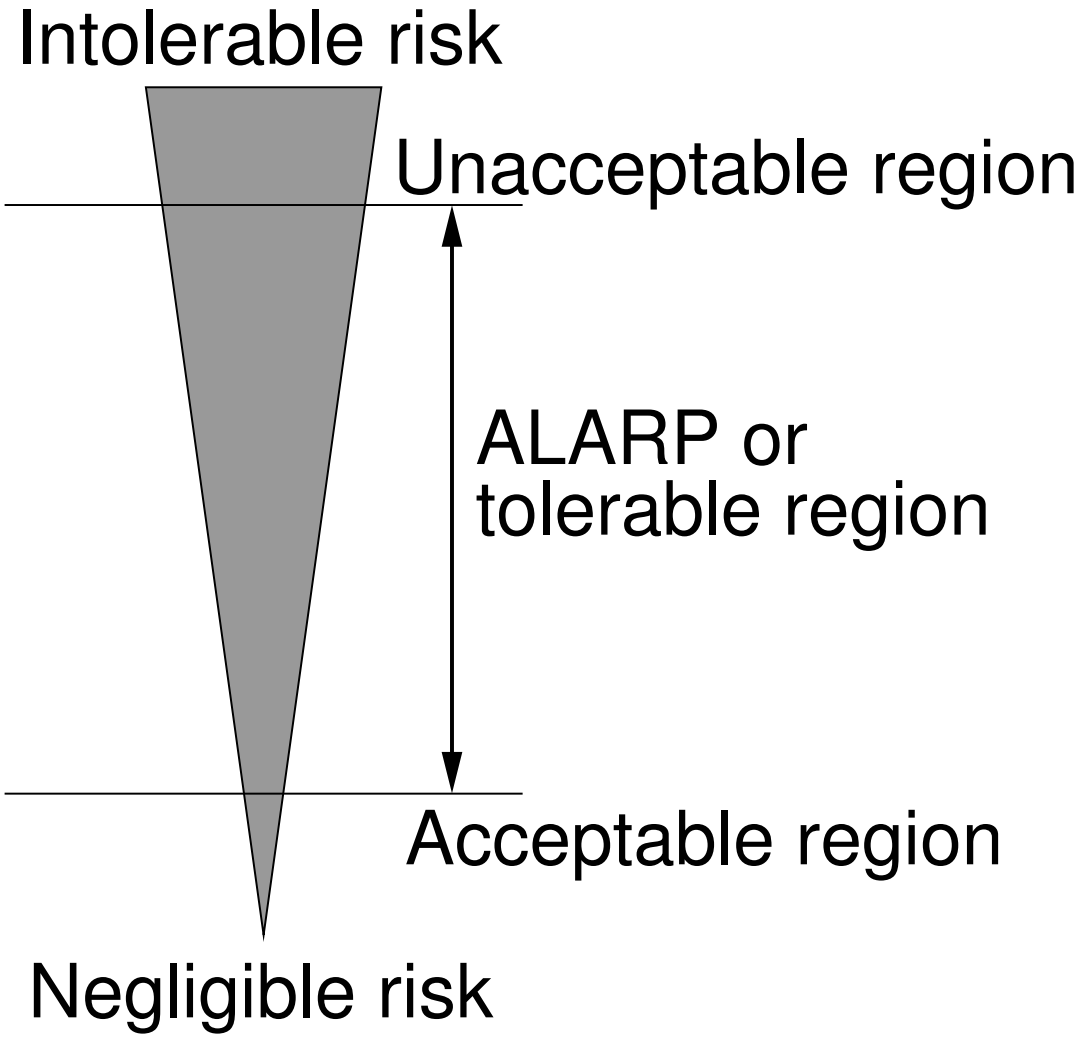
	Consequences			
Frequency	**Catastrophic**	**Critical**	**Marginal**	**Negligible**
Frequent	I	I	I	II
Probable	I	I	II	III
Occasional	I	II	III	III
Remote	II	III	III	IV
Improbable	III	III	IV	IV
Incredible	IV	IV	IV	IV

ACCEPTABILITY OF RISK

- ALARP: As Low As is Reasonably Possible

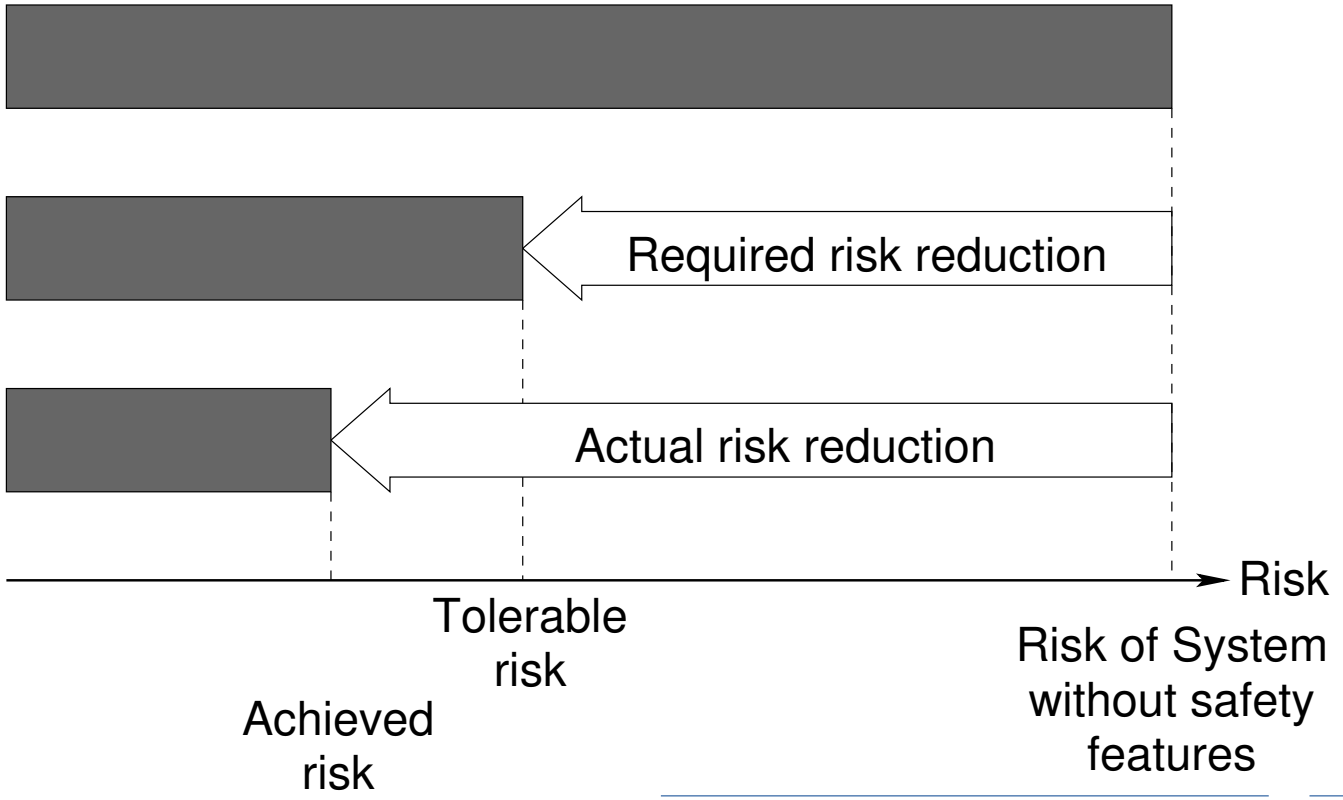
- If risk can easily be reduced, **it should be**
- **Conversely**, a system with significant risk may be acceptable if it offers sufficient benefit and if further reduction of risk is impracticable or incurs unjustifiable cost
- Risk level satisfying ALARP is termed **tolerable risk**

ALARP


Intolerable risk
Unacceptable region
ALARP or
tolerable region
Acceptable region
Negligible risk

RISK MANAGEMENT AND REDUCTION

Producing a safety-critical system can be seen as a process of risk reduction:



ETHICAL CONSIDERATIONS

- Determining risk and more especially acceptability of risk involves morals/judgements/opinions
- Society's view is note asily determined by a clear set of rules - unpredictable - illogical
- Perception: **accidents involving large numbers of deaths** (e.g. Enschede train derailment— 101 deaths) are perceived as being **more serious than smaller accidents** though they may occur less frequently (e.g. German road accident fatalities — approx 250/month)
- Various professional societies offer guidelines on risk management

SAFE VS SECURE

- **Safe**: The system shall … ensure safe operation
- **Secure**: The system shall not … be compromised

  Someone is **intentionally** trying to break it!
  This is also a difficult problem!

- How to develop and maintain systems that can resist malicious attacks intended to damage a computer-based system or its data?

TERMINOLOGY (SECURITY CONCEPTS)

Term	Description
Asset	A system resource that has a value and has to be protected.
Exposure	The possible loss or harm that could result from a successful attack. This can be loss or damage to data or can be loss of time and effort if recovery is necessary after a security breach.
Vulnerability	A weakness in a computer-based system that may be exploited to cause loss or harm.
Attack	An exploitation of a system's vulnerability. Generally, this is from outside the system and is a deliberate attempt to cause some damage.
Threat	Circumstances that have potential to cause loss or harm. You can think of these as a system vulnerability that is subjected to an attack.
Control	A protective measure that reduces a system's vulnerability. Encryption would be an example of a control that reduced a vulnerability of a weak access control system.

ANALOGIES OF SAFETY / SECURITY

- Compare:
  - Exposure vs Accident
  - Vulnerability vs Hazard

CRITICAL SYSTEMS

- Become a domain expert for safety, security or mission criticality
- Become a domain expert for the application domain
- What are the hazards, vulnerabilities, …. ?

- Or have such experts on your team