20
Critical Systems and Software Safety

Distributed Embedded Systems

Philip Koopman

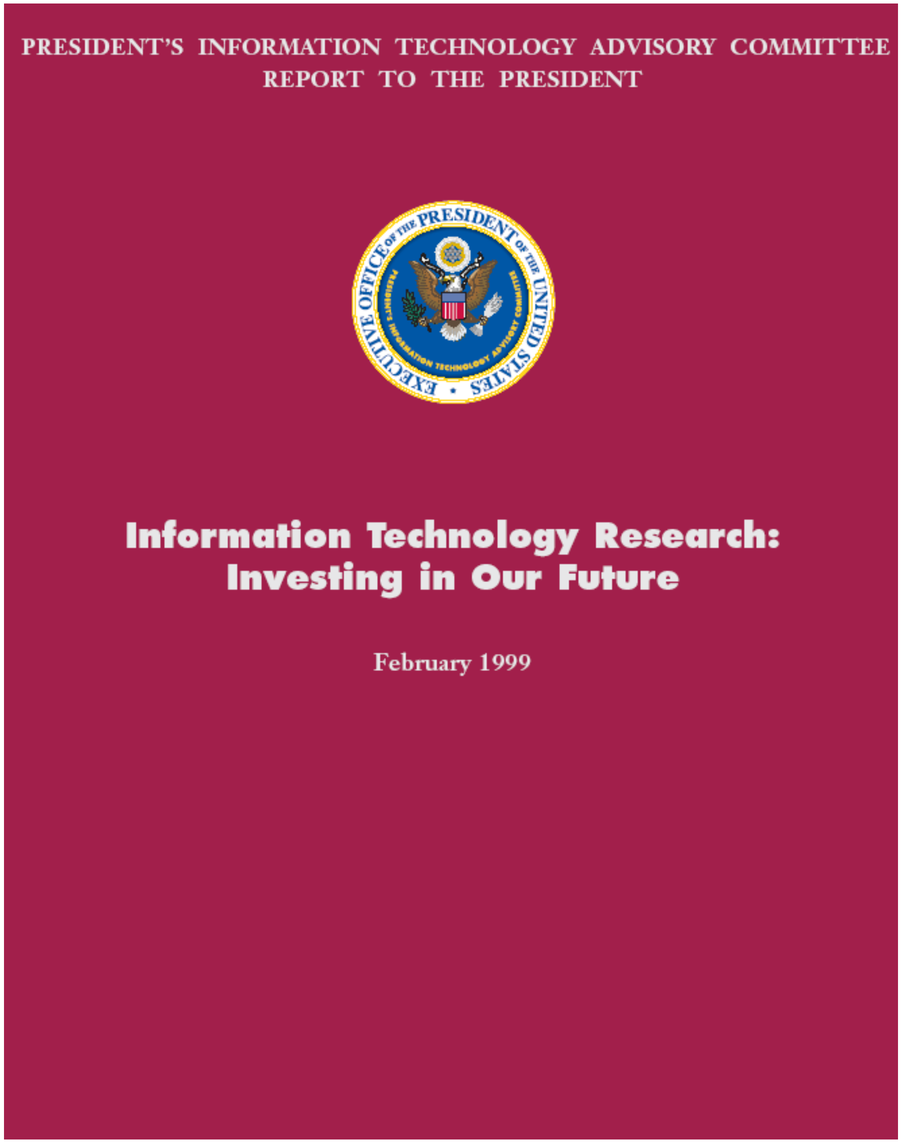
November 16, 2015

Therac 25 Required Reading is testable material. Know the big picture of that story.

Carnegie Mellon

“We have become dangerously dependent on large software systems whose behavior is not well understood and which often fail in unpredictable ways.”

– US President's IT Advisory Committee, February 1999

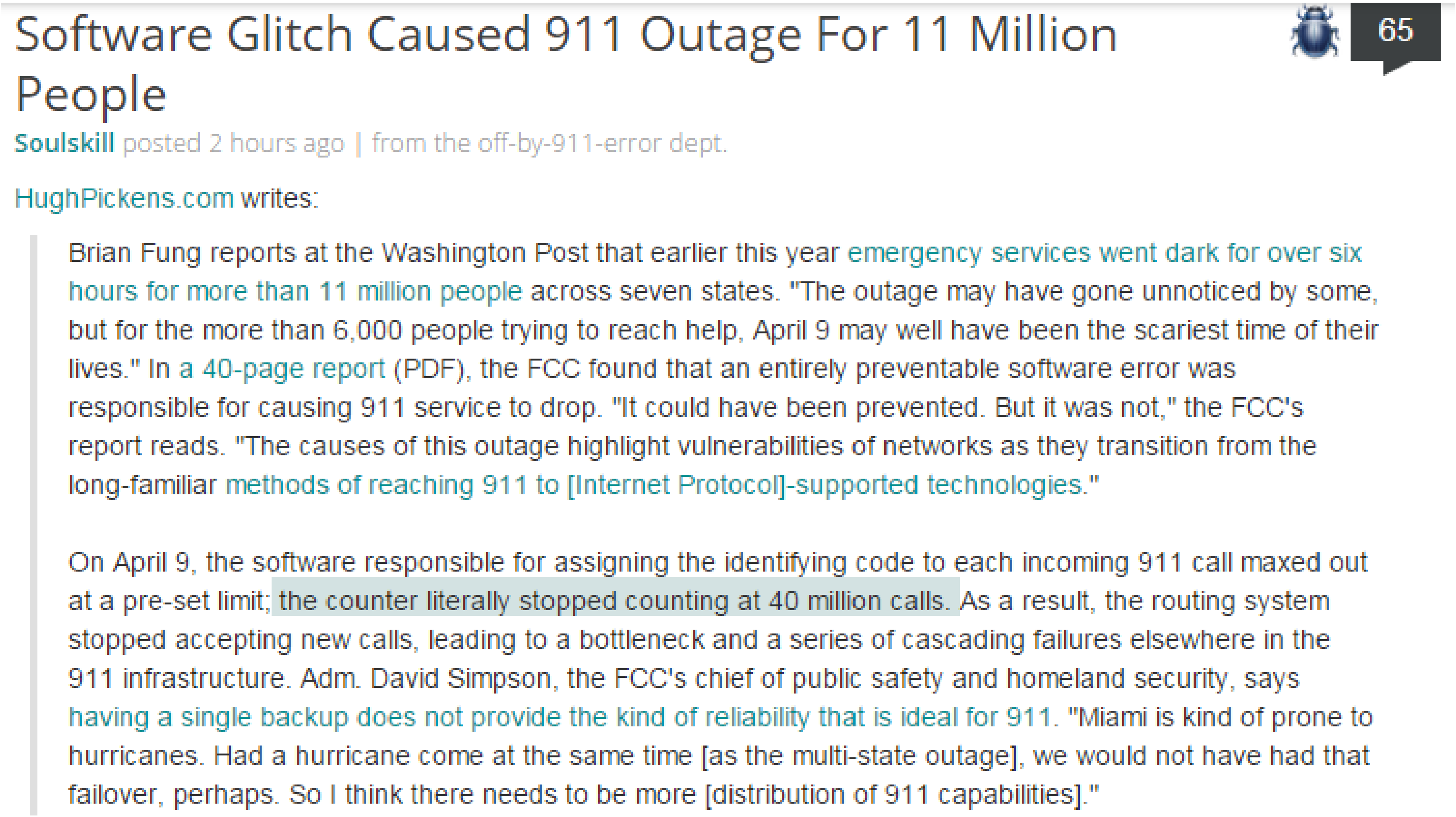


PRESIDENT’S INFORMATION TECHNOLOGY ADVISORY COMMITTEE
REPORT TO THE PRESIDENT

Information Technology Research: Investing in Our Future

February 1999

E-911 Phone Service Outage April 9, 2014



Software Glitch Caused 911 Outage For 11 Million People

Soulskill posted 2 hours ago | from the off-by-911-error dept.

HughPickens.com writes:

> Brian Fung reports at the Washington Post that earlier this year emergency services went dark for over six hours for more than 11 million people across seven states. "The outage may have gone unnoticed by some, but for the more than 6,000 people trying to reach help, April 9 may well have been the scariest time of their lives." In a 40-page report (PDF), the FCC found that an entirely preventable software error was responsible for causing 911 service to drop. "It could have been prevented. But it was not," the FCC's report reads. "The causes of this outage highlight vulnerabilities of networks as they transition from the long-familiar methods of reaching 911 to [Internet Protocol]-supported technologies."
>
> On April 9, the software responsible for assigning the identifying code to each incoming 911 call maxed out at a pre-set limit; the counter literally stopped counting at 40 million calls. As a result, the routing system stopped accepting new calls, leading to a bottleneck and a series of cascading failures elsewhere in the 911 infrastructure. Adm. David Simpson, the FCC's chief of public safety and homeland security, says having a single backup does not provide the kind of reliability that is ideal for 911. "Miami is kind of prone to hurricanes. Had a hurricane come at the same time [as the multi-state outage], we would not have had that failover, perhaps. So I think there needs to be more [distribution of 911 capabilities]."

Preview

- **General safety engineering**
  - Terminology
  - Basic Techniques (FMEA/FTA/HAZOP)
- **Risk Management**
  - PHA matrix
- **Related info (not covered in lecture)**
  - Therac 25 – a cautionary tale
  - How software in a radiation therapy machine killed people
  - Covered in 18-348 and 18-349
    - Should have been covered in whatever embedded course you took!
    - Required, testable reading for this lecture as a refresher

Traditional Safety Engineering

- **Largely based on industrial environments such as chemical plants**

- **Hazards based on uncontrolled *release of energy***
  - Risk was associated with amount of energy and time (*e.g.*, explosion)
  - Risk was reduced via containment, minimizing potential energy in system, supervised operation in risky situations

- **Embedded system engineering has to encompass**
  - Release of energy from controlled system (physical damage)
  - Release of information from controlled system (security)
  - Avoiding inability to release energy/information (reliability/denial of service)

Definitions of Safety

- Informally:

  “Nothing Bad Will Happen”

- N. Leveson, *Safeware*: (pg. 181)

  “Freedom from accidents or losses”

  - But, of course, no system can be completely “safe” in an absolute sense
  - So the issue is how to make something safe enough …
… given limited budget, time, and resources
  - Focuses on end goal of accidents rather than risk

- N. Storey, *Safety-Critical Computer Systems*: (pg. 2)

  “System will not endanger human life or the environment”

  - More emphasis on removing hazards than actual accidents
  - Again, issue is that complete safety is impossible
    - But, that’s not an excuse for slacking on design efforts!

Terminology

- **Hazard:**
  - A situation with potential danger to people, environment, or material
  - Example: interlock that prevents subway door from opening isn’t activated

- **Incident (near miss):**
  - A situation with actual danger to people, environment, or material
  - Sometimes a hazard results in an incident
  - Something that under other circumstances would have been an accident
  - Example: subway door opens, but nobody is leaning against it

- **Accident (also called a mishap):**
  - If you get unlucky, what could be an incident turns into an accident
  - Events that cause death, injury, environmental, or material damage
  - Example: subway door opens, and someone falls out of car

- **Risk:**
  - A combination of probability of hazards, and severity of likely outcomes.
  - (more on this later)

Classical Safety-Critical Failure Handling

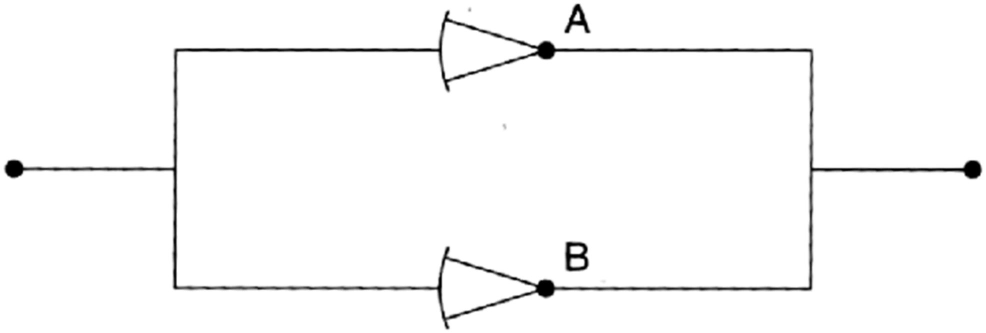
- **Fail Operational**
  - Even though something fails the system keeps working
  - Usually accomplished through redundancy
- **Fail-over to reduced capability system**
  - Simpler algorithms
  - Mechanical backup
  - Person
- **Fail Safe**
  - Identify a safe state and transition to that safe state upon failure
    - Tension between safety and availability; a system with 0% availability might well be 100% safe
  - Sometimes use a reduced capability system as “limp home” to a safe state
- **Key capabilities:**
  - Knowing what will happen when a component fails
  - Designing systems with redundancy to avoid single-point failure vulnerabilities

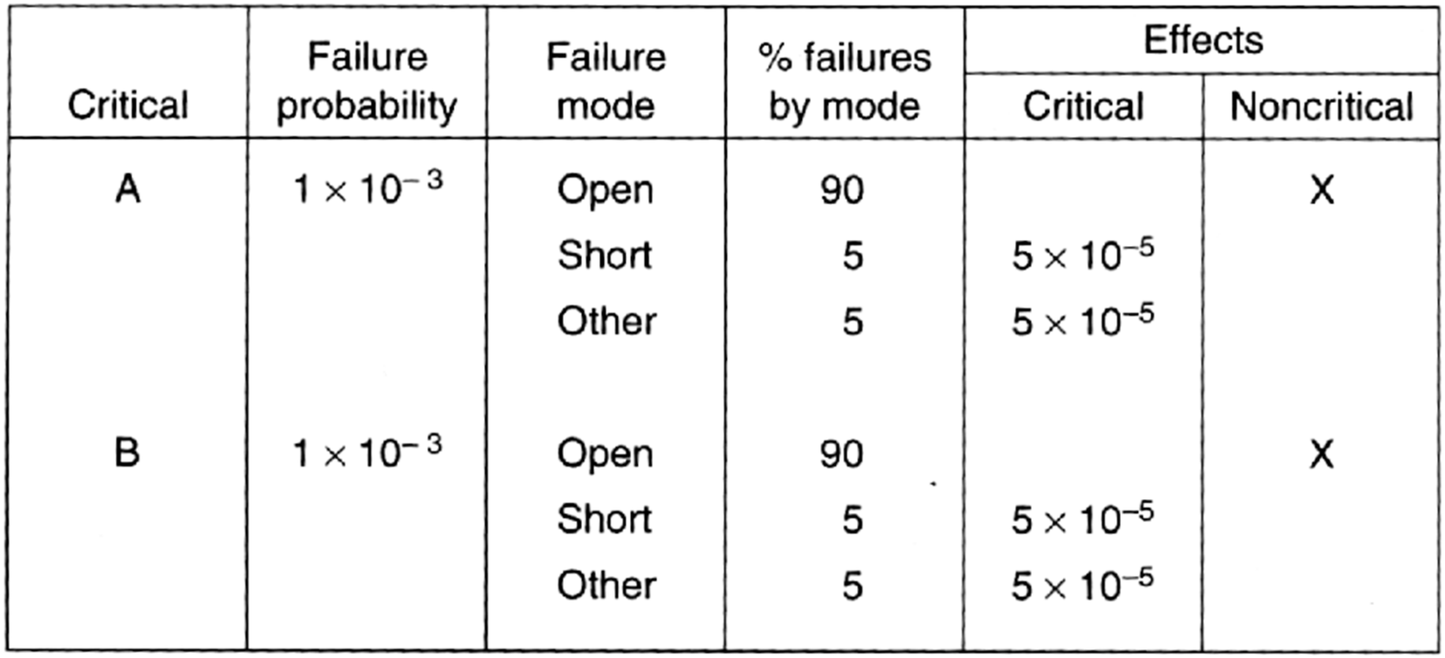
Basic Analysis Technique – FMEA

- **Failure Mode and Effects Analysis (FMEA)**
  - Probably the most commonly used technique in embedded system design
  - Looks for consequences of component failures (forward chaining technique)
  - Limitation: requires expert analysis to decide what to analyze

- **Failure Mode and Effects Criticality Analysis (FMECA)**
  - Similar to FMEA but with two columns added
    - Overall assessment of criticality
    - Possible actions to reduce criticality

- **General goal**
  - Find failures that have high criticality and do something to reduce probability

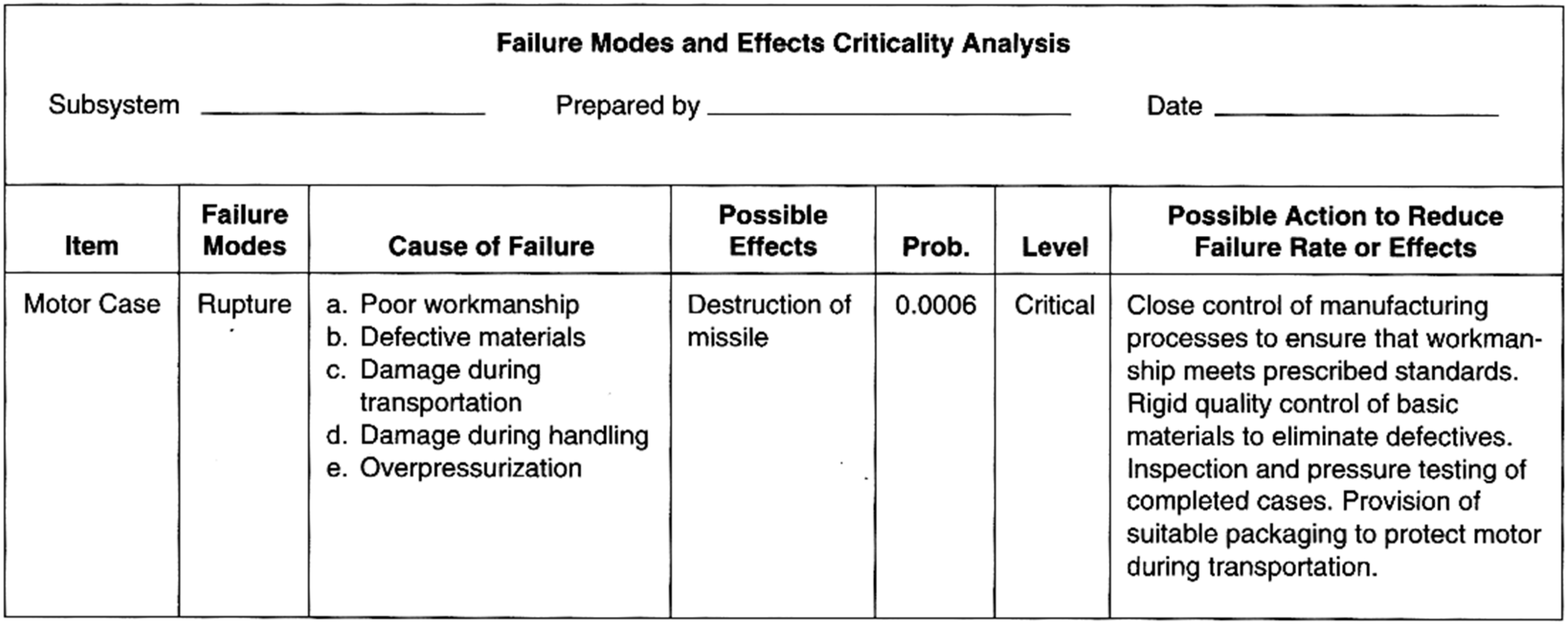




Critical	Failure probability	Failure mode	% failures by mode	Effects	
				Critical	Noncritical
A	1×10^{-3}	Open	90		X
		Short	5	5×10^{-5}	
		Other	5	5×10^{-5}	
B	1×10^{-3}	Open	90		X
		Short	5	5×10^{-5}	
		Other	5	5×10^{-5}	

FIGURE 14.9
FMEA for a system of two amplifiers in parallel. (Source: W. E. Vesely, F. F. Goldberg, N. H. Roberts, and D. F. Haasl, *Fault Tree Handbook*, NUREG-0492, U.S. Nuclear Regulatory Commission, Washington, D.C., 1981, page II-3)

[Leveson]



Failure Modes and Effects Criticality Analysis

Subsystem ________________ Prepared by ______________________ Date ________________

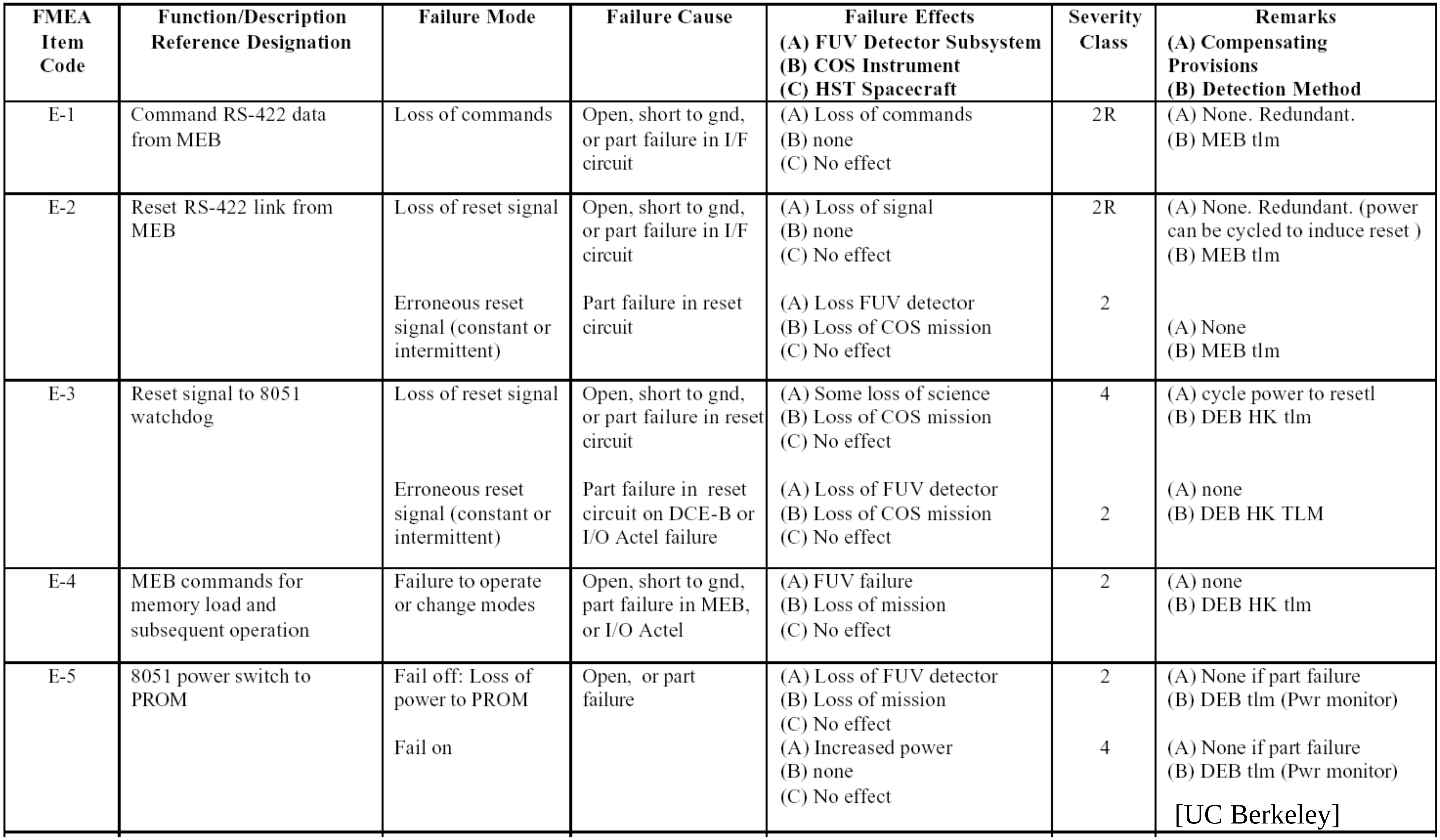
Item	Failure Modes	Cause of Failure	Possible Effects	Prob.	Level	Possible Action to Reduce Failure Rate or Effects
Motor Case	Rupture	a. Poor workmanship b. Defective materials c. Damage during transportation d. Damage during handling e. Overpressurization	Destruction of missile	0.0006	Critical	Close control of manufacturing processes to ensure that workmanship meets prescribed standards. Rigid quality control of basic materials to eliminate defectives. Inspection and pressure testing of completed cases. Provision of suitable packaging to protect motor during transportation.

FIGURE 14.10
A sample FMECA.

[Leveson]

Hubble Far Ultraviolet Instrument Example

- Severity 2 means loss of FUV mission, but not all of HST
- Severity 2R means redundant units would have to fail



FMEA Item Code	Function/Description Reference Designation	Failure Mode	Failure Cause	Failure Effects **(A) FUV Detector Subsystem** **(B) COS Instrument** **(C) HST Spacecraft**	Severity Class	Remarks **(A) Compensating Provisions** **(B) Detection Method**
E-1	Command RS-422 data from MEB	Loss of commands	Open, short to gnd, or part failure in I/F circuit	(A) Loss of commands (B) none (C) No effect	2R	(A) None. Redundant. (B) MEB tlm
E-2	Reset RS-422 link from MEB	Loss of reset signal	Open, short to gnd, or part failure in I/F circuit	(A) Loss of signal (B) none (C) No effect	2R	(A) None. Redundant. (power can be cycled to induce reset) (B) MEB tlm
		Erroneous reset signal (constant or intermittent)	Part failure in reset circuit	(A) Loss FUV detector (B) Loss of COS mission (C) No effect	2	(A) None (B) MEB tlm
E-3	Reset signal to 8051 watchdog	Loss of reset signal	Open, short to gnd, or part failure in reset circuit	(A) Some loss of science (B) Loss of COS mission (C) No effect	4	(A) cycle power to resetl (B) DEB HK tlm
		Erroneous reset signal (constant or intermittent)	Part failure in reset circuit on DCE-B or I/O Actel failure	(A) Loss of FUV detector (B) Loss of COS mission (C) No effect	2	(A) none (B) DEB HK TLM
E-4	MEB commands for memory load and subsequent operation	Failure to operate or change modes	Open, short to gnd, part failure in MEB, or I/O Actel	(A) FUV failure (B) Loss of mission (C) No effect	2	(A) none (B) DEB HK tlm
E-5	8051 power switch to PROM	Fail off: Loss of power to PROM	Open, or part failure	(A) Loss of FUV detector (B) Loss of mission (C) No effect	2	(A) None if part failure (B) DEB tlm (Pwr monitor)
		Fail on		(A) Increased power (B) none (C) No effect	4	(A) None if part failure (B) DEB tlm (Pwr monitor)

[UC Berkeley]

Basic Analysis Technique – FTA

- **Fault Tree Analysis (FTA)**
  - Origins: 1961 to ensure no accidental Minuteman missile launches
  - Analyzes possible causes of hazards, but you already have to have the list of hazards to begin with (backward chaining technique)
  - Problems:
    - Doesn’t represent real-time issues
    - Doesn’t represent system state or operating modes

- **General goal**
  - Eliminate single-point vulnerabilities
  - Corresponds to making sure each likely failure is protected by an “AND” gate

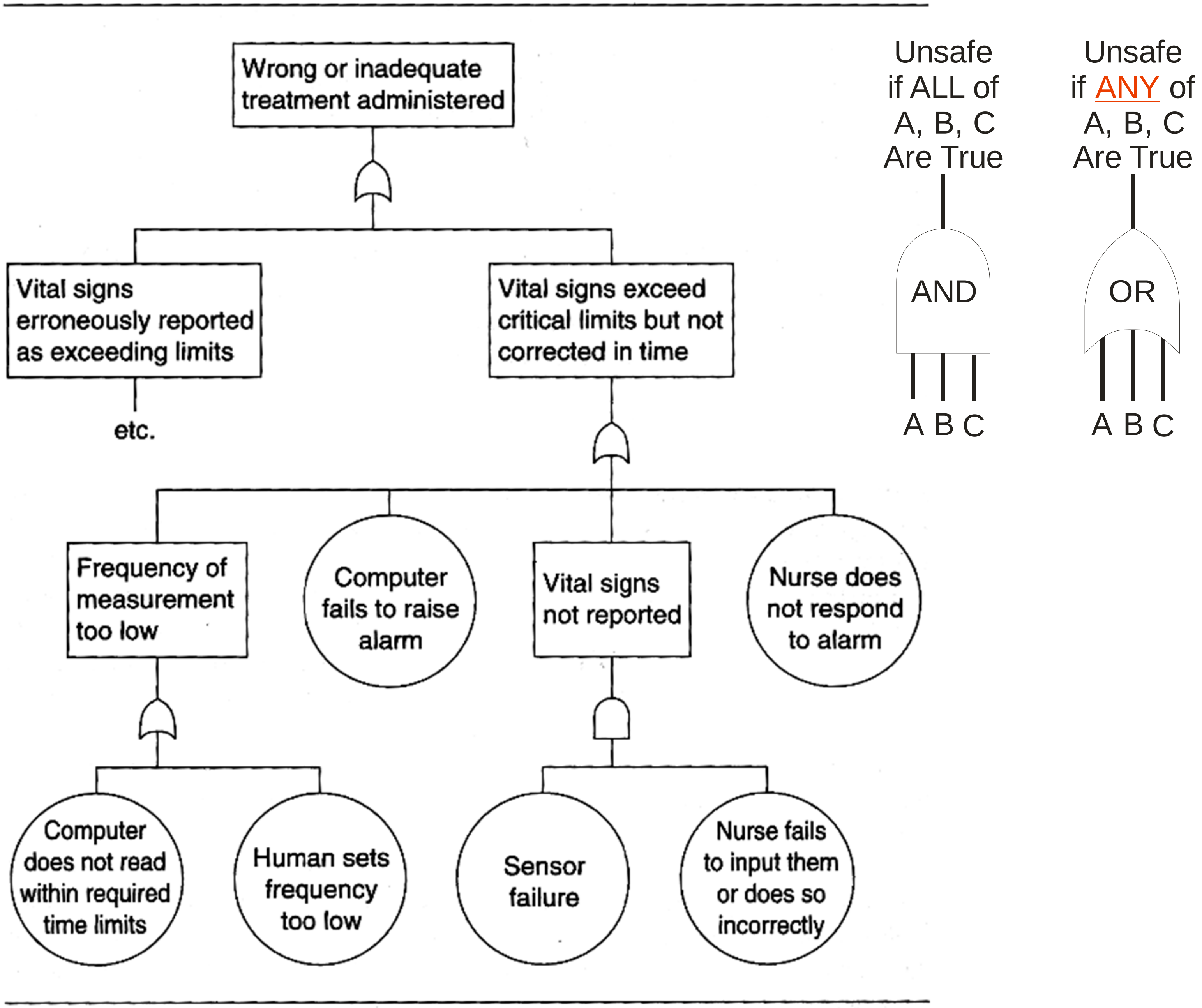
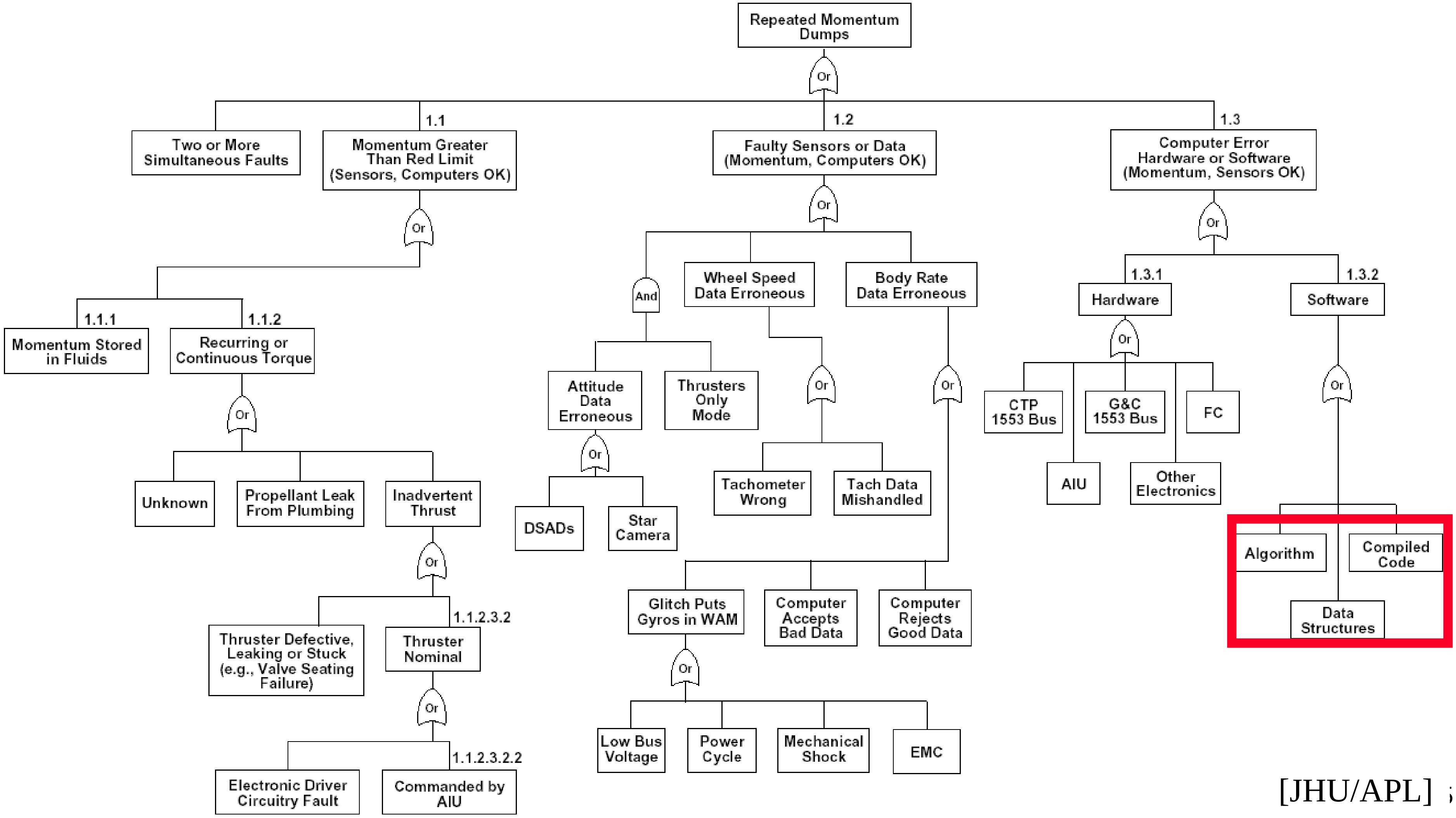

Wrong or inadequate treatment administered
Vital signs erroneously reported as exceeding limits
etc.
Vital signs exceed critical limits but not corrected in time
Frequency of measurement too low
Computer fails to raise alarm
Vital signs not reported
Nurse does not respond to alarm
Computer does not read within required time limits
Human sets frequency too low
Sensor failure
Nurse fails to input them or does so incorrectly
Unsafe if ALL of A, B, C Are True
AND
A B C
Unsafe if ANY of A, B, C Are True
OR
A B C

FIGURE 14.3
Portion of a fault tree for a patient monitoring system.

[Leveson]

NEAR Spacecraft Fault Tree

- **NASA Near Earth Asteroid Rendezvous mission**
  - 20 December 1998; used up most of thruster fuel
  - Too-small limit on lateral accelerometer + defective command recovery script

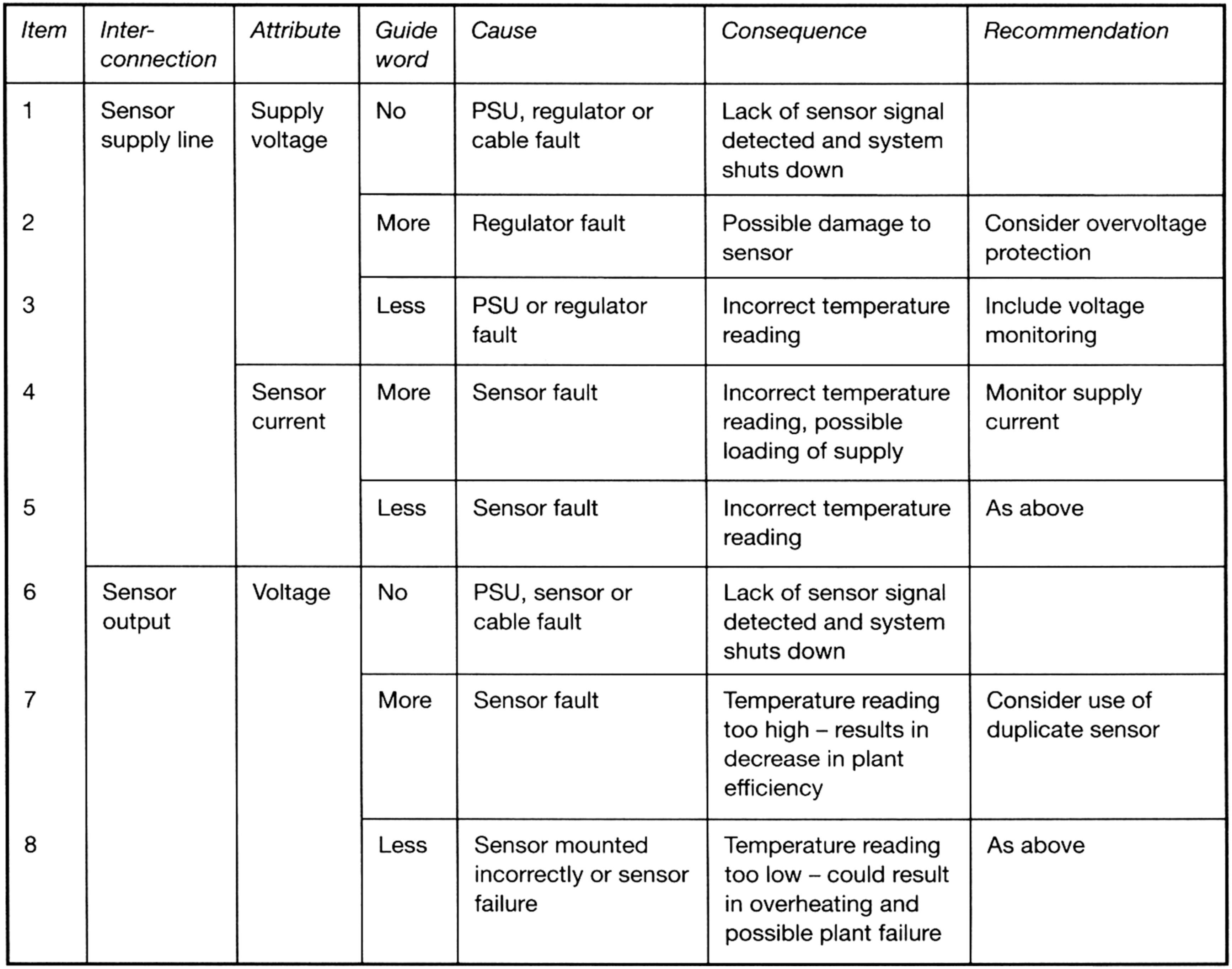

Repeated Momentum Dumps
Or
Two or More Simultaneous Faults
1.1
Momentum Greater Than Red Limit (Sensors, Computers OK)
Or
1.1.1
Momentum Stored in Fluids
1.1.2
Recurring or Continuous Torque
Or
Unknown
Propellant Leak From Plumbing
Inadvertent Thrust
Or
Thruster Defective, Leaking or Stuck (e.g., Valve Seating Failure)
1.1.2.3.2
Thruster Nominal
Or
Electronic Driver Circuitry Fault
1.1.2.3.2.2
Commanded by AIU
1.2
Faulty Sensors or Data (Momentum, Computers OK)
Or
And
Attitude Data Erroneous
Or
DSADs
Star Camera
Thrusters Only Mode
Wheel Speed Data Erroneous
Or
Tachometer Wrong
Tach Data Mishandled
Body Rate Data Erroneous
Or
Glitch Puts Gyros in WAM
Or
Low Bus Voltage
Power Cycle
Mechanical Shock
EMC
Computer Accepts Bad Data
Computer Rejects Good Data
1.3
Computer Error Hardware or Software (Momentum, Sensors OK)
Or
1.3.1
Hardware
Or
CTP 1553 Bus
AIU
G&C 1553 Bus
Other Electronics
FC
1.3.2
Software
Or
Algorithm
Compiled Code
Data Structures

[JHU/APL]

HAZOP: Hazard & Operability Studies

- **Origins in 1960s for chemical & process control plants**
  - Focus on interconnections & interactions between components
  - Flow of materials or energy

- **Uses “guide words” in specification to trigger analysis items**
  - “no” – what if there is no flow?
  - “more” – what if a limit is exceeded?
  - “as well as” – what if something additional happens?
  - “part of” – what if something only partially completes?
  - “reverse” – what if flow is reversed?
  - “other than” – something else happens, e.g., incorrect data
  - “early” – signal earlier than deadline window
  - “late” – signal later than deadline window
  - “before” – out of order; arrives early
  - “after” – out of order; arrives late



Item	*Inter-connection*	*Attribute*	*Guide word*	*Cause*	*Consequence*	*Recommendation*
1	Sensor supply line	Supply voltage	No	PSU, regulator or cable fault	Lack of sensor signal detected and system shuts down	
2			More	Regulator fault	Possible damage to sensor	Consider overvoltage protection
3			Less	PSU or regulator fault	Incorrect temperature reading	Include voltage monitoring
4		Sensor current	More	Sensor fault	Incorrect temperature reading, possible loading of supply	Monitor supply current
5			Less	Sensor fault	Incorrect temperature reading	As above
6	Sensor output	Voltage	No	PSU, sensor or cable fault	Lack of sensor signal detected and system shuts down	
7			More	Sensor fault	Temperature reading too high – results in decrease in plant efficiency	Consider use of duplicate sensor
8			Less	Sensor mounted incorrectly or sensor failure	Temperature reading too low – could result in overheating and possible plant failure	As above

Figure 3.5 Part of a simplified HAZOP results table for a temperature sensor. [Storey]

Embedded Distributed System Failures

- **In addition to all the above, there can be network problems**
  - Network failures can be attacked by using replicated networks
  - Network packet errors due to noise are a problem

- **Be sure to calculate effects of dropped network packets!**
  - Contributing causes of lost packets:
    - High bit error rate
    - Noise bursts due to electric motor operation
    - Retries not supported in order to simplify scheduling
    - Collision-based communication protocols
  - Event triggered systems – loss of packet can leave system in incorrect state
  - Time triggered systems – repeated loss of packet can cause loss of control loop stability
    - It doesn’t take many lost packets to lead to problems in a large scale fleet

- **Then there are failures due to COTS software…**

What’s Risk?

- **Risk = penalty * likelihood**
  - Penalty can be in dollars, lives, injuries, amount deadline missed by
  - Likelihood is probability that a particular hazard will be “activated” and result in an undesirable outcome
  - The product can be considered as an expected value of cost to project

- **“Normal” risks can be prioritized as weighted factors in project risk**

$$Project\ Risk = \sum_i \left(Cost_i \cdot Probability_i \right)$$

  - “Pareto Ranking” used to identify largest risk contributors
    - Pareto rule is the 80/20 rule -- 80% of the problems are from 20% of the risks
    - Pareto ranking simply means address the top of the ranked list first

- **But rare+costly events are a problem**
  - How big is infinite penalty multiplied by near-zero likelihood?
  - Catastrophic penalties often must be guarded against even for near-zero probability

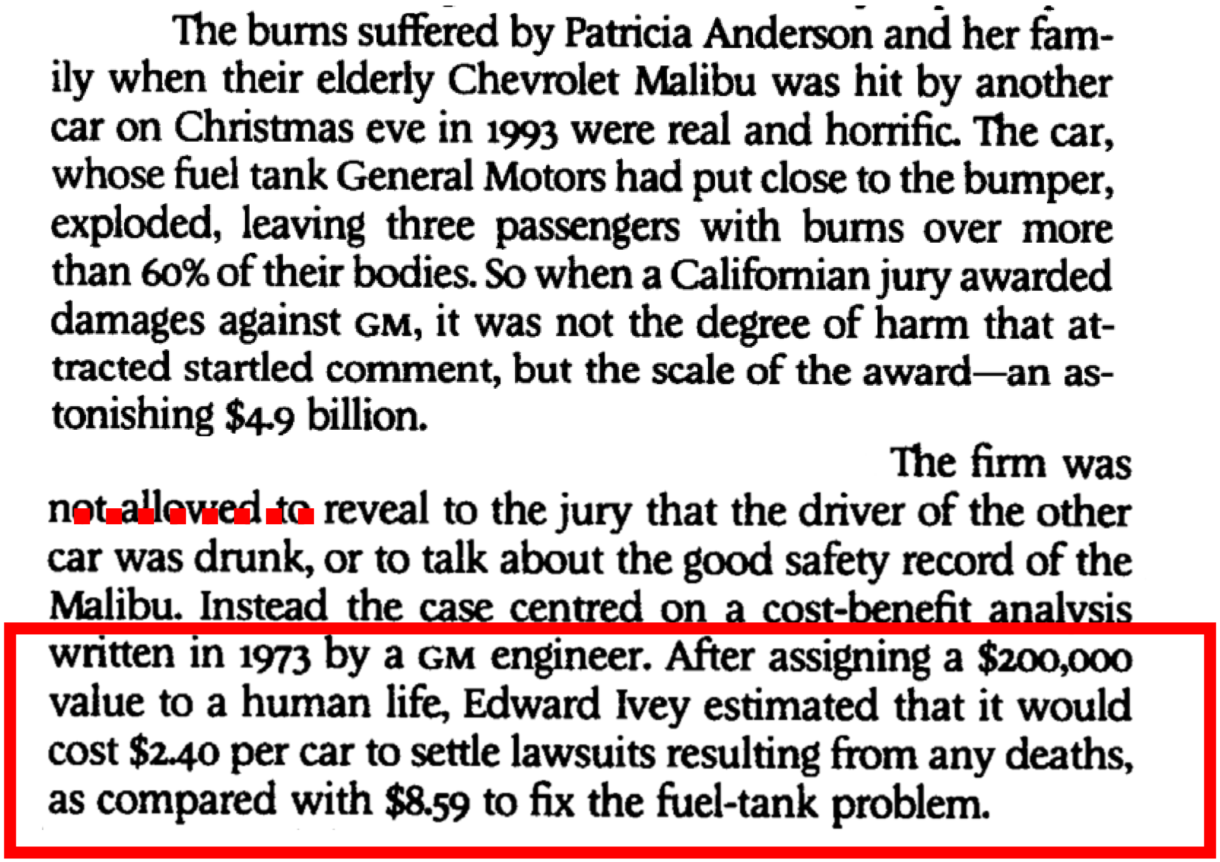
Risk Prioritization – PHA

- **Mathematically, RISK = PROBABILITY * CONSEQUENCE**
  - This is mathematically neat and tidy; it is an "expected cost" function
- **Typical Risk Ranking Matrix: (caution: use with care!)**
  - Sometimes these tables are asymmetric, with higher weight on consequence
  - Often used as part of a PHA (Preliminary Hazard Analysis) to focus efforts
  - BTW – also a nice way to prioritize bug reports for non-critical code

EXAMPLE **RISK**		**Probability**				
		Very High	**High**	**Medium**	**Low**	**Very Low**
Conse-quence	**Very High**	**Very High**	**Very High**	**Very High**	**High**	**High**
	High	**Very High**	**High**	**High**	**Medium**	**Medium**
	Medium	**High**	**High**	**Medium**	**Medium**	**Low**
	Low	**High**	**Medium**	**Medium**	**Low**	**Very Low**
	Very Low	**Medium**	**Low**	**Low**	**Very Low**	**Very Low**

Putting A $$$ Amount On Human Life Is Risky

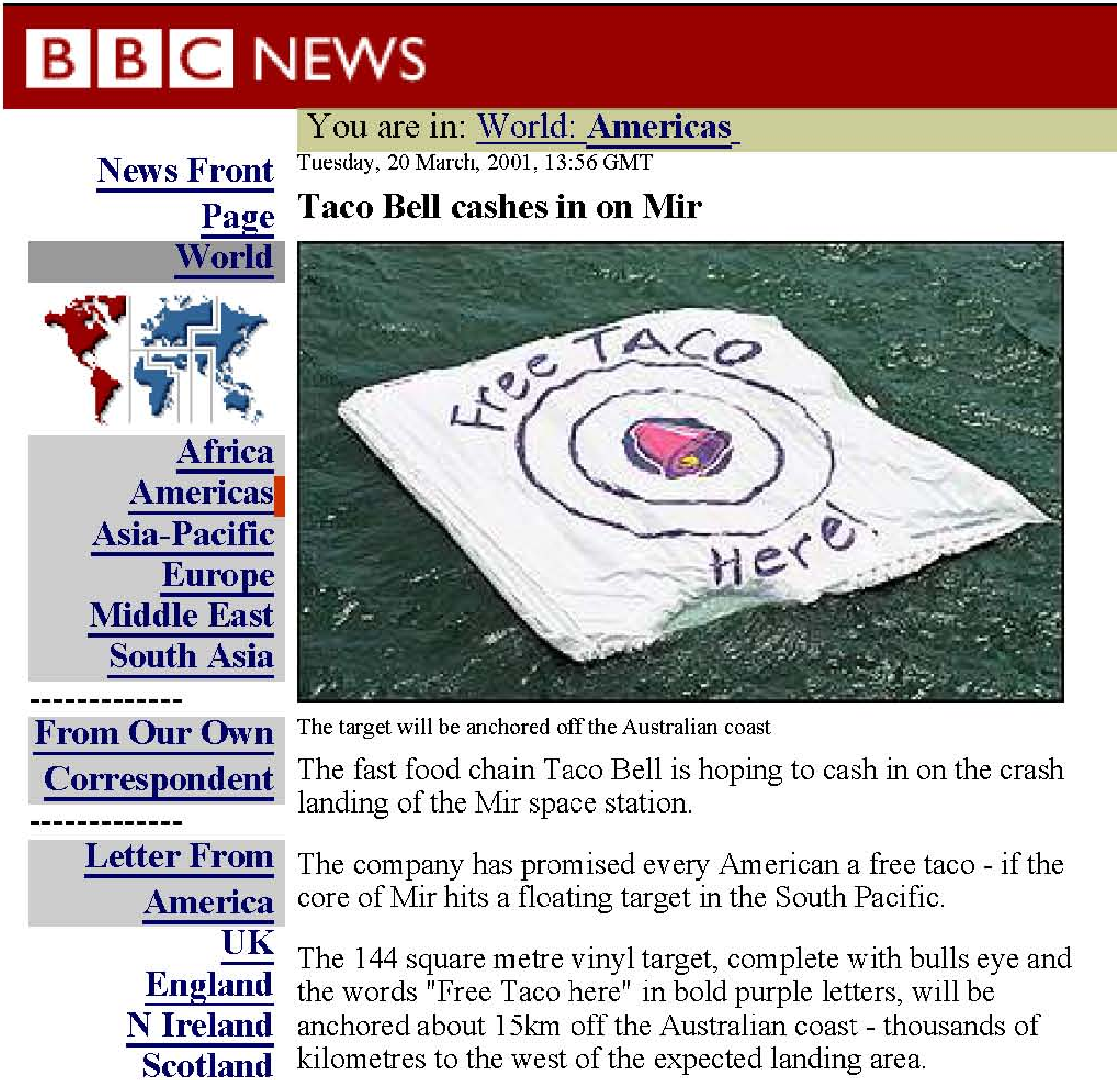
- “GM and the Law,” *The Economist*, July 17, 1999



The burns suffered by Patricia Anderson and her family when their elderly Chevrolet Malibu was hit by another car on Christmas eve in 1993 were real and horrific. The car, whose fuel tank General Motors had put close to the bumper, exploded, leaving three passengers with burns over more than 60% of their bodies. So when a Californian jury awarded damages against GM, it was not the degree of harm that attracted startled comment, but the scale of the award—an astonishing $4.9 billion.

The firm was not allowed to reveal to the jury that the driver of the other car was drunk, or to talk about the good safety record of the Malibu. Instead the case centred on a cost-benefit analysis written in 1973 by a GM engineer. After assigning a $200,000 value to a human life, Edward Ivey estimated that it would cost $2.40 per car to settle lawsuits resulting from any deaths, as compared with $8.59 to fix the fuel-tank problem.

- **Very likely the award amount was reduced**
  - But, having a written record of quantified, accepted risk creates problems
  - A cost-benefit justification document played a role in the Ford Pinto gas tank story



BBC NEWS

You are in: World: Americas

News Front Page
World

Africa
Americas
Asia-Pacific
Europe
Middle East
South Asia

From Our Own Correspondent

Letter From America
UK
England
N Ireland
Scotland

Tuesday, 20 March, 2001, 13:56 GMT

Taco Bell cashes in on Mir

The target will be anchored off the Australian coast

The fast food chain Taco Bell is hoping to cash in on the crash landing of the Mir space station.

The company has promised every American a free taco - if the core of Mir hits a floating target in the South Pacific.

The 144 square metre vinyl target, complete with bulls eye and the words "Free Taco here" in bold purple letters, will be anchored about 15km off the Australian coast - thousands of kilometres to the west of the expected landing area.

Risk Management

- **Identify risks and track them**
  - "Unknown risks" that haven't been thought of are of course a problem

- **Can Mitigate, Avoid, or Accept Risks**
  - Mitigate – perform a risk reduction task
    - Understand risk well enough to decide on avoid/accept choice
    - Perform research to solve problem
    - Obtain insurance to collapse risk probability into a known fixed cost -- especially for rare but expensive risk items
  - Avoid
    - Use a less risky approach (can't always do this)
  - Accept
    - Decide that expected cost (probability * impact) is not worth reducing further
    - Often the right choice when avoiding the risk costs more than expected risk cost
      - (But, see ethics lecture when costs affect other people)
  - Ignore
    - Proceed ahead blindly – uninformed acceptance
    - Seldom ideal; sometimes unavoidable when in a hurry

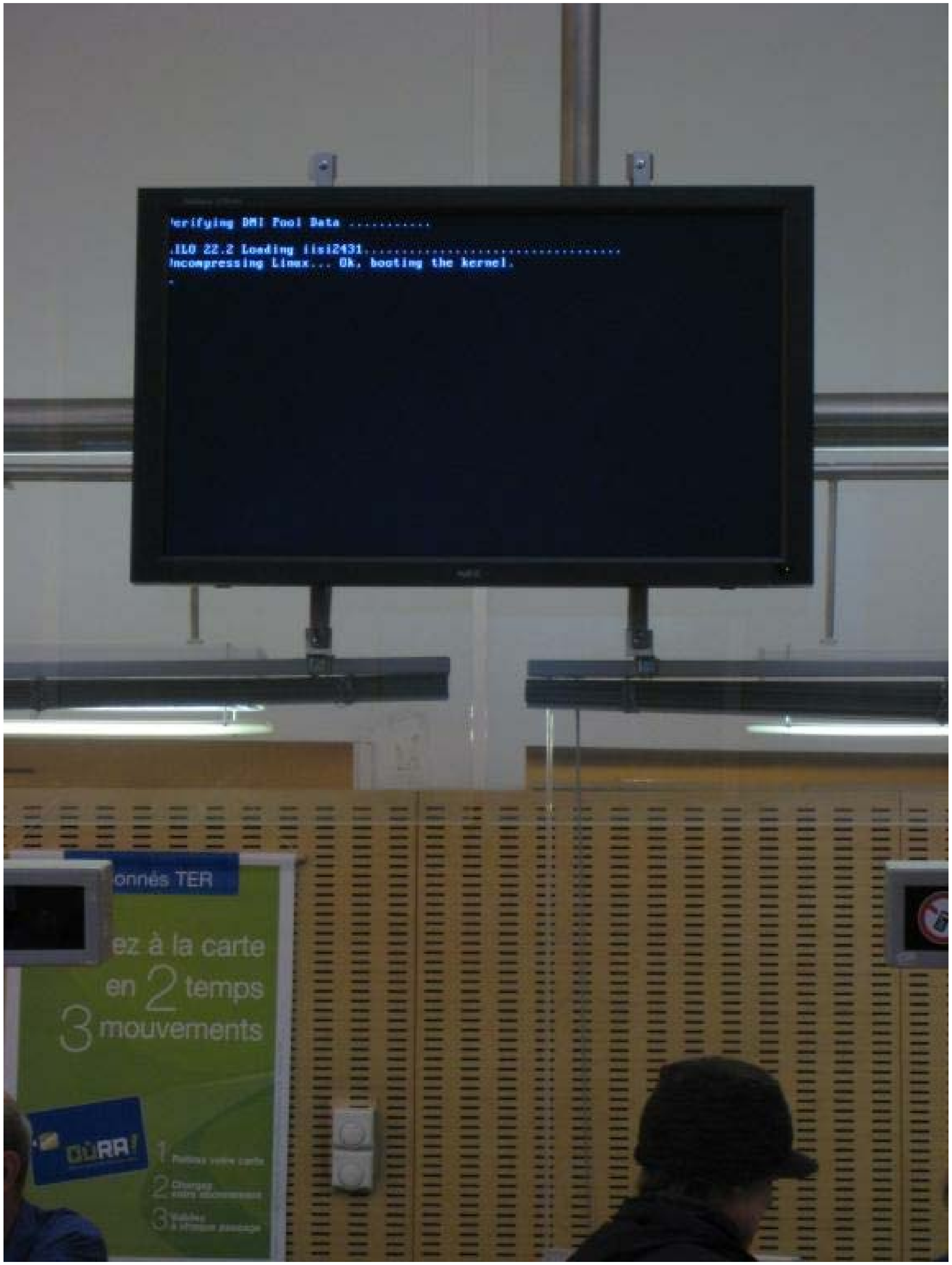
Civil Aircraft Hazard Categories

- **Catastrophic** (10^{-9}/hr)
  - Prevents continued safe flight and landing
- **Hazardous** (10^{-7}/hr)
  - Large reduction in safety margins; perhaps fatal injuries to some passengers
- **Major** (10^{-5}/hr)
  - Significant reduction in safety margins; perhaps non-fatal injuries to passengers
- **Minor** (10^{-3}/hr)
  - Slight reduction in safety margins; reasonable workarounds
- **Nuisance** (10^{-2}/hr)
  - Failure causes no effect

- **Notes:**
  - You can’t achieve better than about 10^{-5}/hr without good redundancy
  - Increase in crew workload is a significant factor to be considered
  - Related fact: risk from lightning is
    - 5×10^{-7} deaths per person-year = 5×10^{-11} /hr
    - 1.25×10^{-6} injuries per person-year
    - (There is often a tacit argument that death rates lower than this are acceptable)

Grenoble Train Station
January 2007

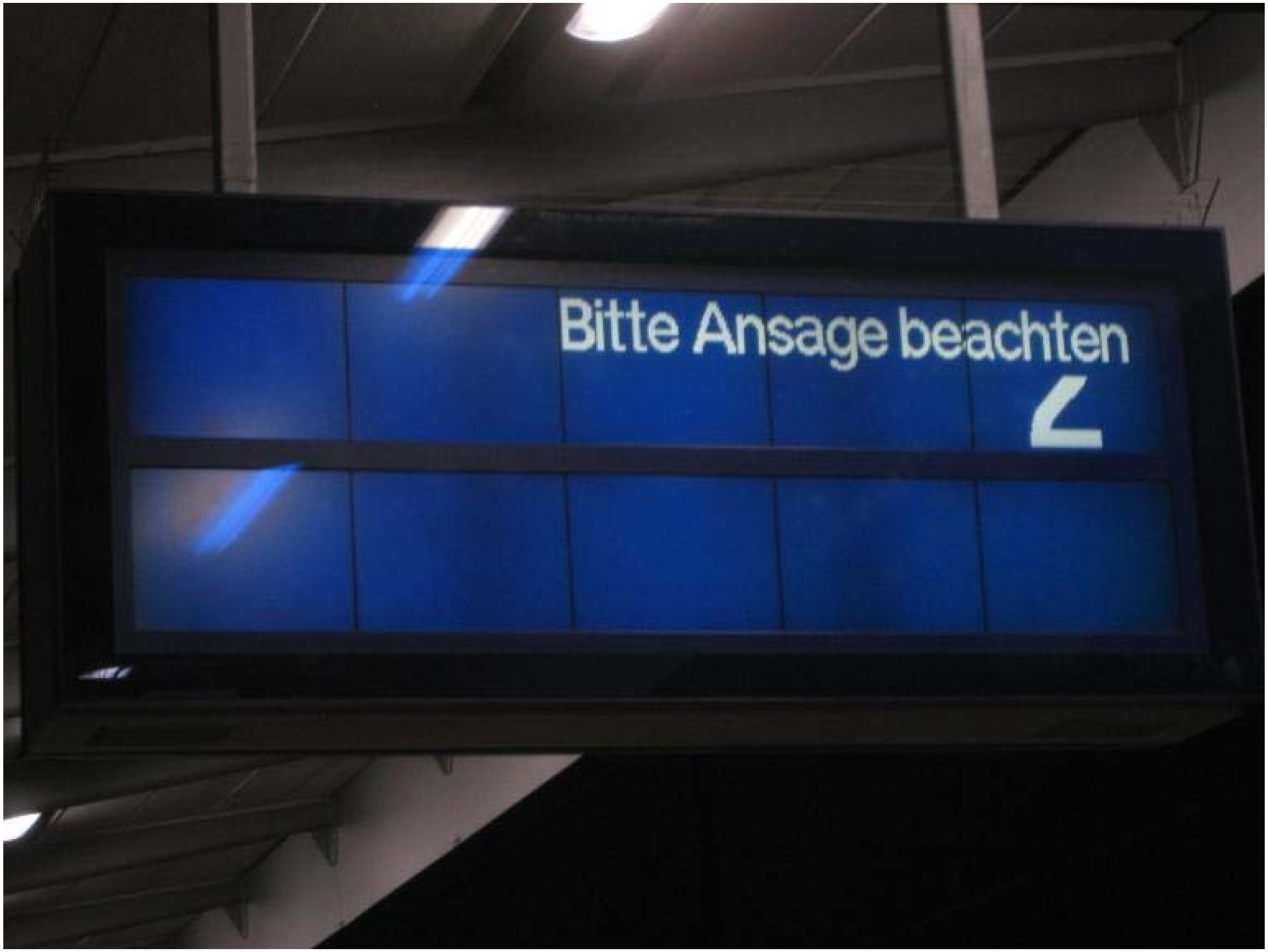
(Our train leaves in 3 minutes. Which track is it on?)



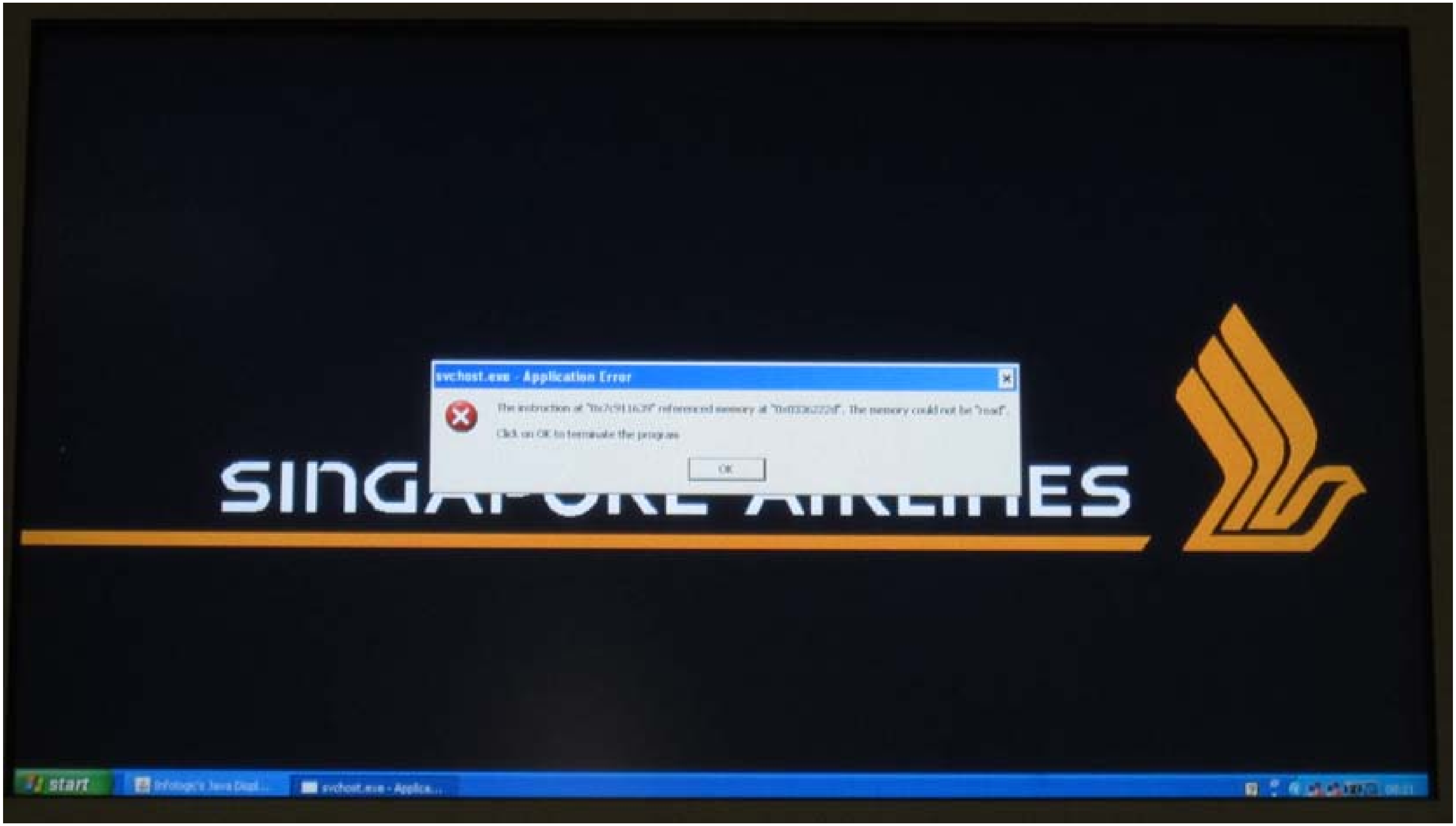
```
'erifying DMI Pool Data ...........

.ILO 22.2 Loading lisi2431....................................
'ncompressing Linux... Ok, booting the kernel.
.
```

Essen Germany Train Station March 2010

(“Is This The Train To Dusseldorf?”)



Departure Status Display (not on an aircraft)

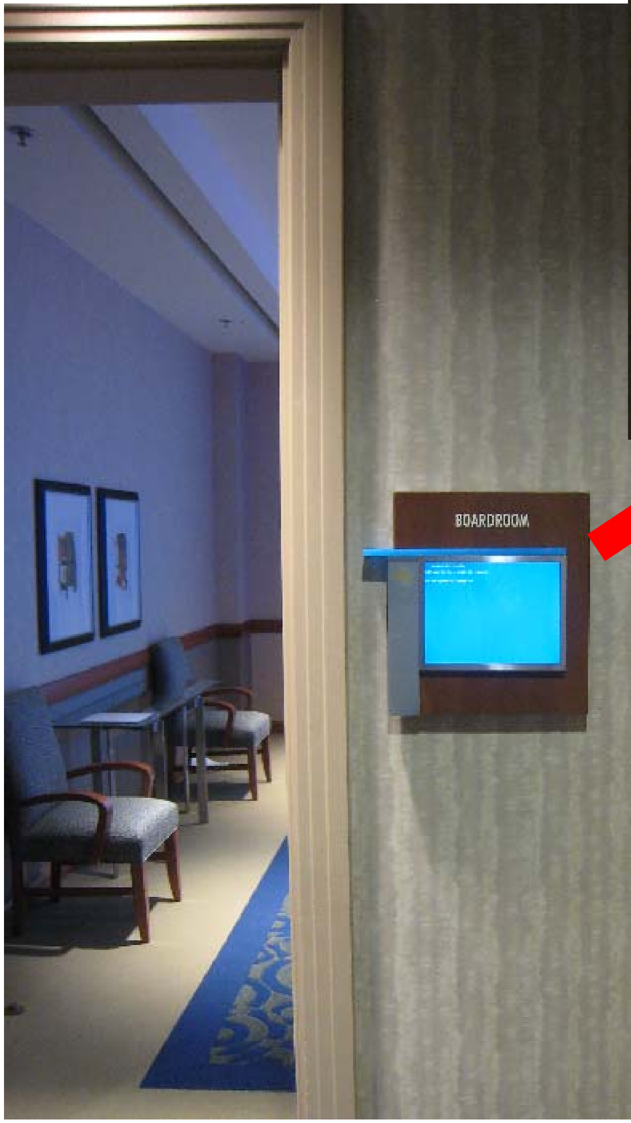


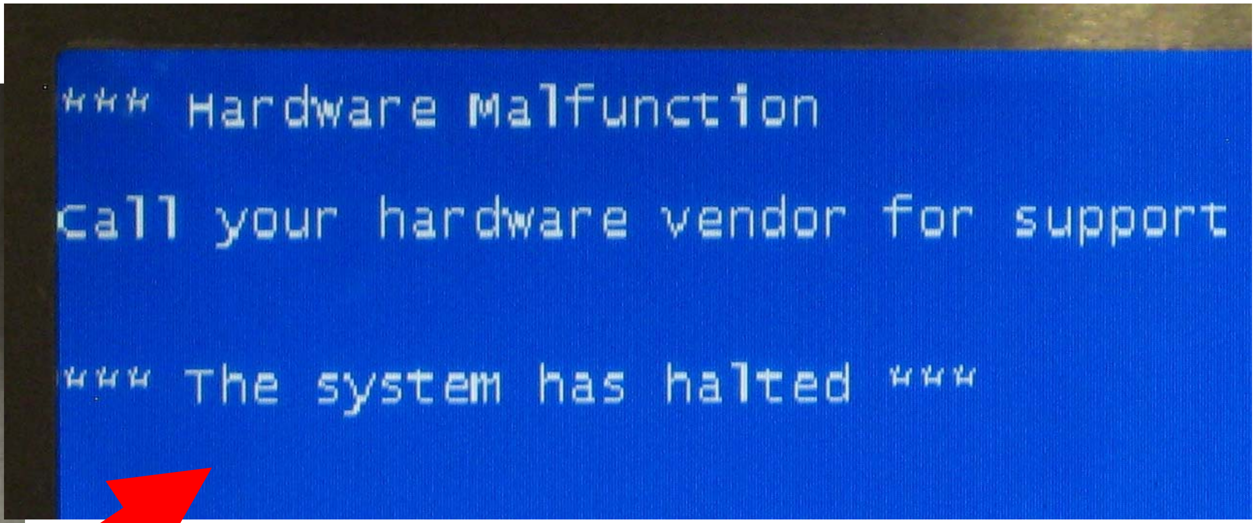
svchost.exe - Application Error

The instruction at "0x7c911639" referenced memory at "0x033b222d". The memory could not be "read".

Click on OK to terminate the program

2008 Workshop on Exception Handling, Atlanta GA



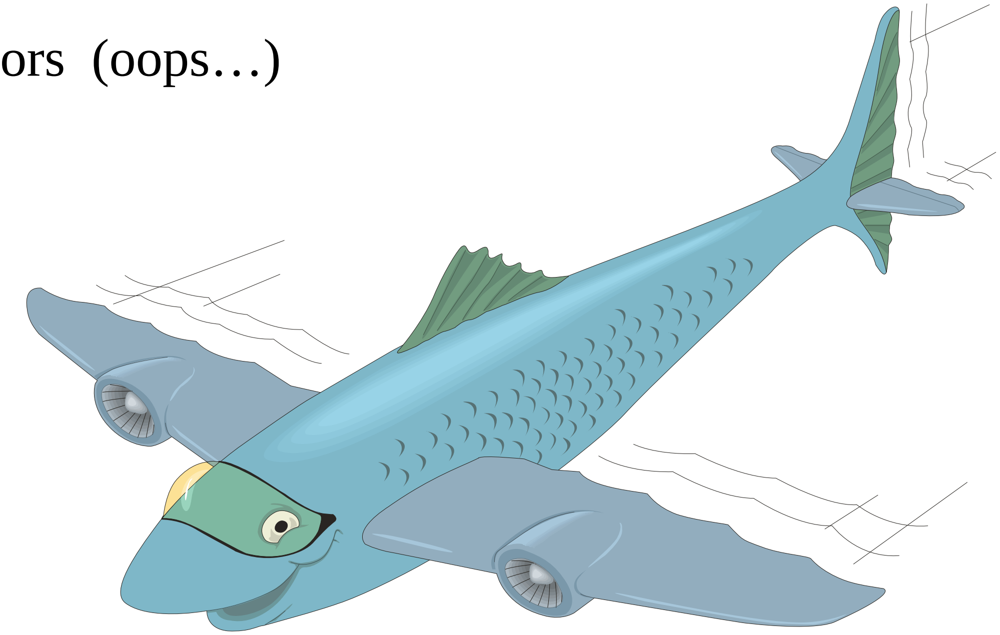


```
*** Hardware Malfunction

call your hardware vendor for support

*** The system has halted ***
```

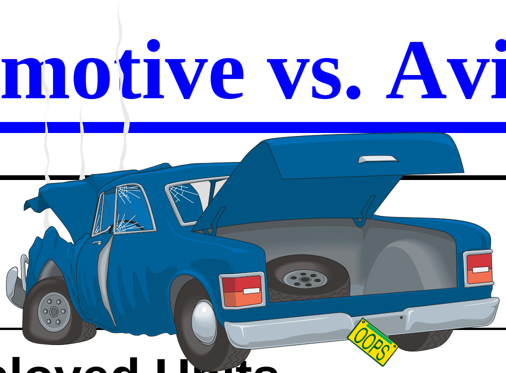
Why Not Build Cars Like Aircraft or Trains?

- **We all “know” that flying is safer than driving**
  - (This is only true per mile, not per hour)

- **So, use commercial aircraft techniques to build automated vehicles**
  - Computer-controlled navigation & tactical maneuvers
  - Redundant hardware
  - Near-perfect software
  - High-quality design and components
  - Highly trained professional operators (oops…)



Automotive vs. Aviation Safety

	U.S. Automobiles	U.S. Commercial Aircraft
Deployed Units	~100,000,000	~10,000
Operating hours/year	~30,000 Million	~55 Million
Passengers	1-5+	50-500+
Cost per vehicle	~$20,000	~$65 Million
Vehicle Mass	4,000 pounds	500,000 pounds
Mortalities/year	42,000	~120
Accidents/year	21 Million	~170
Mortalities / Million Hours	0.71	2.1
Operator/Maintenance skill	variable	High
Redundancy Levels	Brakes only (will change soon)	All flight-critical systems

- Aviation autopilot is probably easier than an automotive autopilot

Why Not Aerospace Approaches For Cars?

- **Based on culture of redundant HW, perfect SW**
- **Too expensive**
  - Component “Pain threshold” for vehicles is at the $.05 level
  - Higher levels of cost OK for Europe if they provide performance value
- **Different operating environment/reaction time**
- **Difficult to enforce maintenance**
  - People run out of gas & engine oil; ignore “idiot lights”
  - Aircraft don’t leave gate if something is broken
  - End-of-life wearout -- old vehicles stay on the road
  - Can we ensure same maintenance quality?

- **Poorly trained operators**
  - Yearly driver exam with road test?
  - Required simulator time for accident response?

The Safety Case

- **A safety case is:**
  A well-reasoned argument proving that the system is safe
  - One nice technique is an annotated fault-tree called GSN (Goal Structure Notation)

- **Safety cases are a relatively new idea for software**
  - Exactly what goes in a safety case is still a research topic
  - But, one way to build one is:

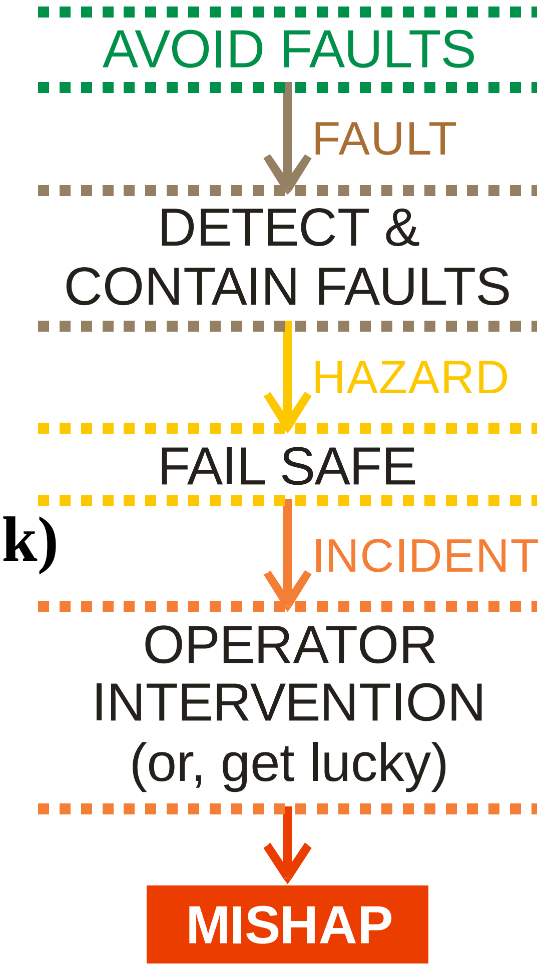
1. Perform a HAZOP to identify hazards
2. Rank hazards to identify the ones that are important enough to address
3. Use a fault tree (FTA) to represent the sources of hazards
4. Demonstrate sufficient protection
   (e.g., no single-point failure branches exist in the fault tree)

- **Difficult to accept, but true situation:**
  - We don’t know how to decide if an artifact, taken in isolation, is “safe” in many cases
  - Safety is assured by combination of:
    - Using a good process & managing/reviewing the process
    - Following safety standards written largely in blood of past mistakes
    - Using good analysis techniques to ensure nothing obvious is missed

Fault Mitigation Strategies For Safety

- **Avoid faults occurring**
  - Careful design of software to avoid defects
  - Use robust hardware
- **Detect and contain faults**
  - Error correction HW, redundant CPUs
  - Watchdog timers for failed tasks, etc.
- **Use Fail Safe strategies to mitigate hazards**
  - For example, automatic instant shutdown mechanisms
- **Incidents require operator intervention (or luck)**
  - Operator may be able to react correctly and quickly
  - Incident will be a mishap some fraction of time
- **Want to avoid escalation as much as possible**
  - E.g., fail safe approaches that work to avoid incidents

Each mitigation level attempts to prevent escalation to next level:



(Points based on Leveson 1986, pp. 149-150)

[Hollnagel93]

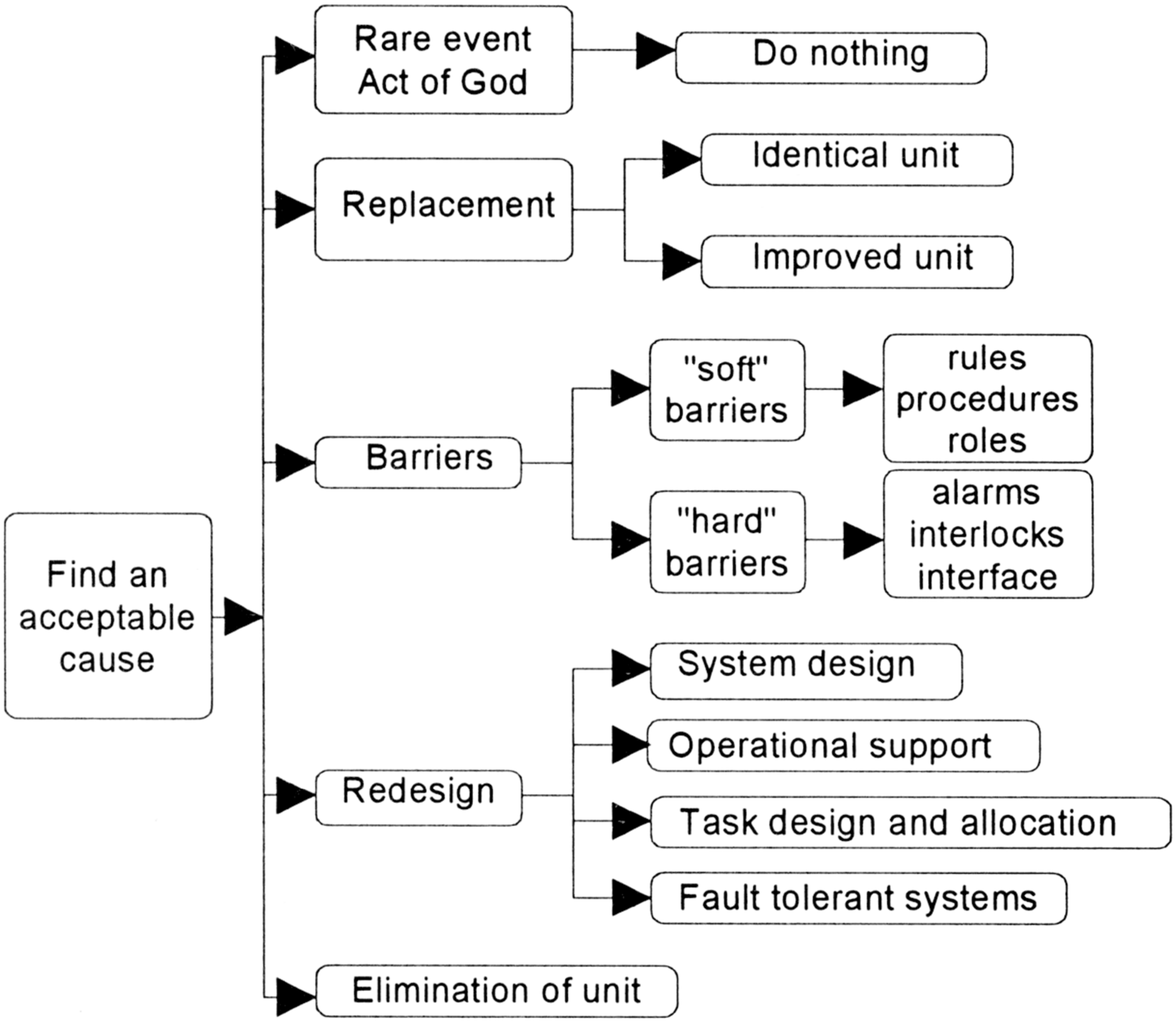
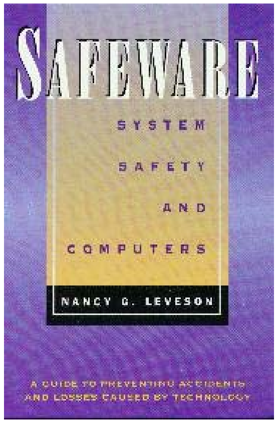


Figure 1-6: Typical reactions to failures and accidents.

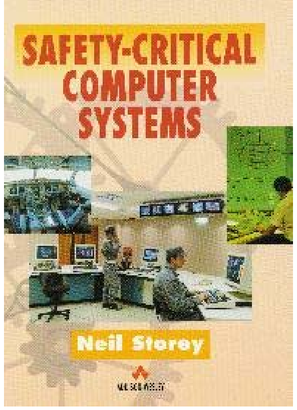
Techniques To Make Systems Safer

- **Address risk explicitly; don’t bury the issue**
  - No system is perfect; but you should address the issue
  - Use a good process
    - A good process enables (***but does not guarantee***) a good product
- **Design system at an appropriate level of reliability/availability/safety**
  - Commonly, this means looking for single-point failures
  - Isolate safety-critical portions of system and apply more attention to them
  - Simplicity is a virtue
  - Plan for the unexpected (exceptions)
  - Be competent at writing good software (why do we need to even say this?)
- **Perform verification/validation/certification**
  - Reviews/inspections
  - Testing (but, you can’t test long enough to ensure ultra-reliability)
  - Formal methods
  - Hazard analysis
- **Include people as part of the system safety design**

Best Safety Books


SAFEWARE
SYSTEM SAFETY AND COMPUTERS
NANCY G. LEVESON
A GUIDE TO PREVENTING ACCIDENTS AND LOSSES CAUSED BY TECHNOLOGY

- **Leveson: *Safeware*, 1995**
  - Gives the big picture on software safety
  - Good discussions of issues & historical stories
  - But, not very specific on how to solve all the problems



- **Storey: *Safety-Critical Computer Systems*, 1996**
  - Roll up your sleeves and get things done
  - Textbook that covers all the basic techniques
  - Doesn’t have a lot of software answers
    (they aren’t really known now, and they weren’t in 1996 either)

Lessons That *Should Have Been* Learned By Now

- **In carefully designed systems, most catastrophes are caused by multiple interacting events**
  - BUT, if that doesn't mean simple things don't happen in everyday systems!
  - Calling anything an "Act of God" usually leads to problems
  - So does blaming the human as the default strategy for avoiding change
- **Just because you can get away with it doesn't make it safe**
  - Challenger O-ring problem was in part based on an escalation of what they (thought they) could get away with
  - Just because a component worked last time doesn't mean it is safe in the new application
- **Humans are part of the system**
  - Operators can help the system or make things worse
  - Often managers are too quick to blame humans for problems that are really due to over-complicated systems
  - Often the cause of a mishap is rooted in the operating environment/***culture***

China Rail Accidents

- **July 2011 – Dozens killed; hundreds injured in China high speed rail accident**
  - One explanation was lightning strike of signaling systems (which are supposed to be safe even if hit by lightning)
  - “The system "failed to turn the green light into red", said An Lusheng, head of the Shanghai Railway Bureau.”

AP

Emergency workers and people work to help passengers from the wreckage of train after two carriages from a high-speed train derailed and fell off a bridge in Wenzhou in east China's Zhejiang province.

Lessons From Therac-25

Adapted with permission from slides by:

Prof. Priya Narasimhan

Assistant Professor of ECE and ISRI

Carnegie Mellon University

Electrical&Computer ENGINEERING

Impact on Life of Failing Software

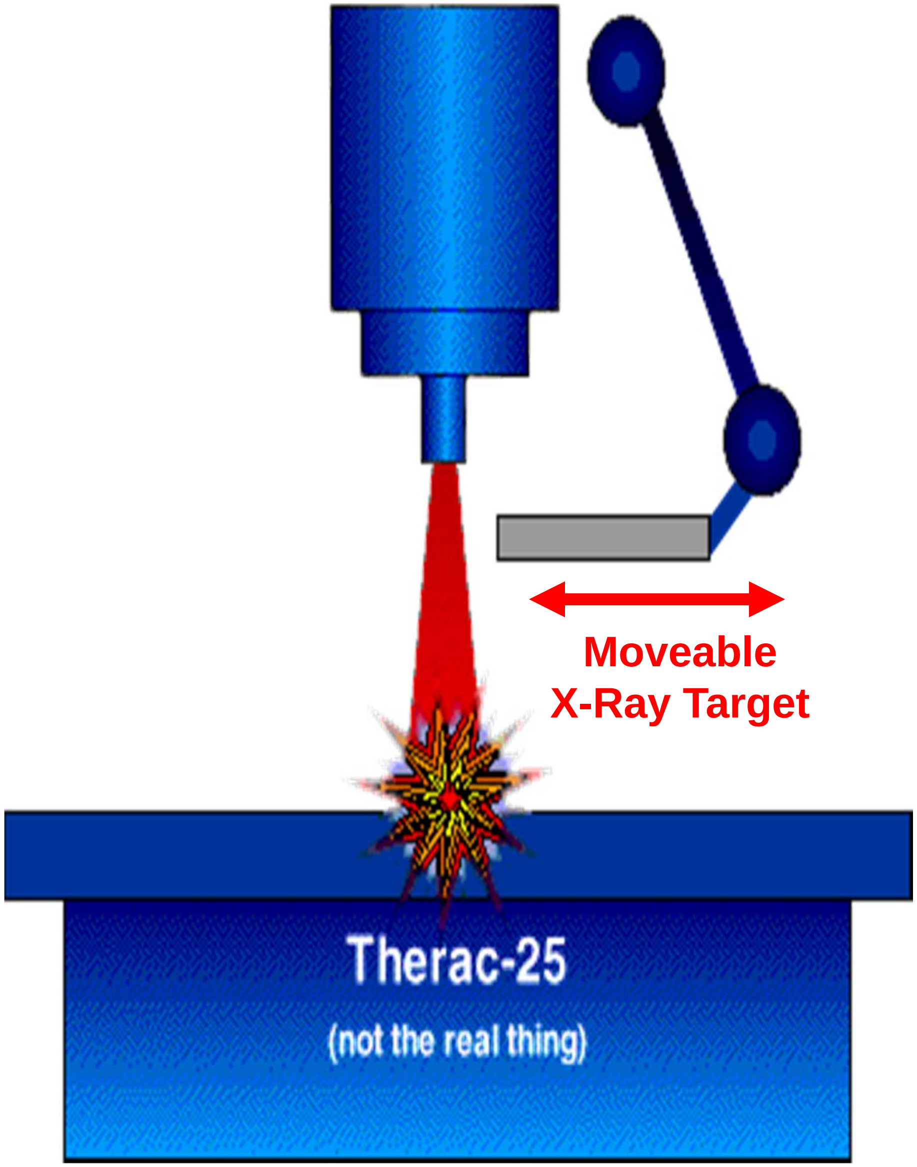
- **Mission-critical applications**
  - Air traffic control
  - Nuclear power plants
  - Aircraft/automobile control

- **Therac-25 radiation failure**
  - One of the most well-documented failures in a safety-critical computer system

- **Why should you care?**
  - Very well documented safety failure case
  - Lessons learned
  - Failures you can relate to, and learn from
  - These problems are not limited to the medical industry – they are generic problems in safety engineering and software engineering

Therac-25 Radiation Therapy Machine

- **Cancer cells are vulnerable to radiation**
  - Radiation kills cancer cells more easily than it kills normal cells
  - So, give just right amount of radiation in a specific target area to only kill cancer cells
- **When a patient undergoes radiation therapy for cancer**
  - Patient is scheduled for several sessions over a period of few weeks
  - Patient told to expect minor discomfort
  - Patient told to expect a “mild sunburn-like” sensation in the treated area
- **Therac-25**
  - Device that targeted electron or X-ray beams on cancerous tissue
  - Purpose was to destroy cancerous cells
  - Deployed from 1983-1987



Why Was the Therac-25 Different?

- **Accelerator that moved to more complete computer control**
  - Think of this as radiation-treatment-by-wire
  - Previous system had software, but also had mechanical safeties

- **Why move to more computer control?**
  - Allowed operators to set up the machine more quickly
  - Gave operators more time to speak with patients
  - Made it possible to treat more patients in a day
    - Machines like this are expensive! More patients/day reduces per patient cost

- **What was the consequence of this decision for system design?**
  - Most of the safety checks during machine operation were in software
  - Hardware safety interlocks were removed
  - Operators were told that there were “so many safety mechanisms” and that it was “virtually impossible” to overdose a patient

What Happened?

- **Massive radiation overdoses in at least six documented cases**
  - Led to deaths and disabilities in patients from radiation overdose

- **These overdoses were due to a combination of**
  - Simple programming errors
  - Inadequate safety engineering
  - Poor human-computer-interaction design

- **What is the point here?**
  - Make you think about and analyze the design and use of software in safety-critical applications
  - What design decisions led to the accidents?
  - How might a different software design have helped to avoid the accidents or minimize the harmful impact?

Predecessors of the Therac-25

- **Predecessor products were Therac-6 and Therac-20**
  - Limited software functionality
  - Used a PDP-11 computer
  - Software written by one person in assembly language
  - Therac-20 had independent protective circuits for monitoring the beam
  - Therac-20 had mechanical interlocks for ensuring safe operation
  - History of safe clinical use without computer control
    - The computer did not directly control release of energy …

      … "release of energy" implies safety is an issue

PDP-11 [Wikipedia]

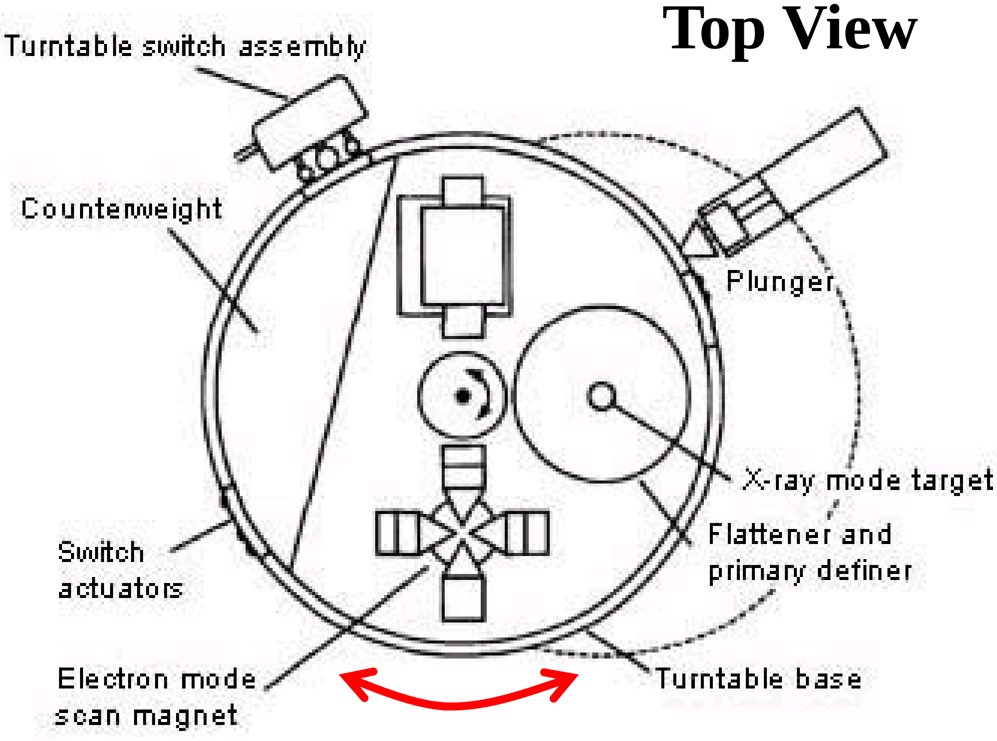
- **And then came the Therac-25 …………..**
  - Influenced by Therac-6 and Therac-20
  - But some different design choices were made

And Then Came the Therac-25

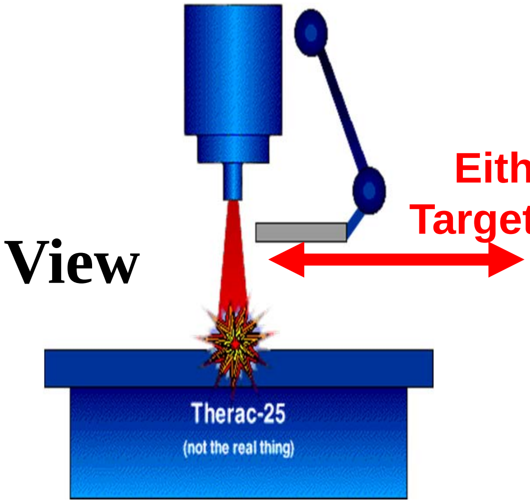
- **Also used a PDP-11 computer (like its predecessors)**
  - Manufacturer decided to take advantage of the computer to control and monitor the hardware
  - Decided not to duplicate all of the hardware interlocks and safety mechanisms
  - Some of the same Therac-6 and Therac-20 software was used

- **Computer was responsible for checking that the turntable (where patient was placed) was in the right position**
  - Positions – X-ray (deep tissue), electron-beam (upper tissue) and field-light (no dosage)
  - 100 times greater beam strength for X-ray therapy (attenuated ~100x if X-ray mode target is correctly in place)

Top View

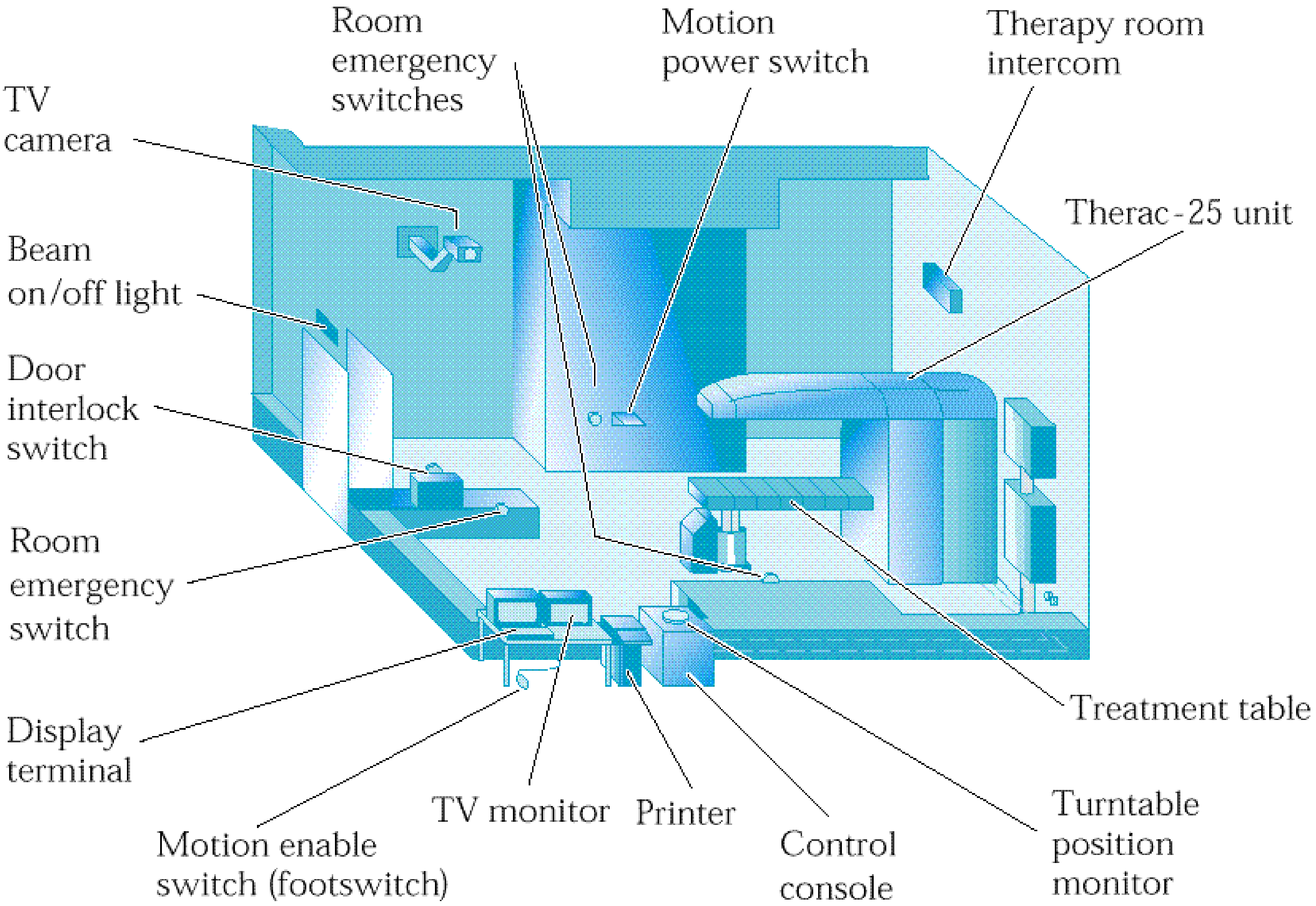

Turntable switch assembly
Counterweight
Plunger
X-ray mode target
Flattener and primary definer
Switch actuators
Electron mode scan magnet
Turntable base

Side View



Either X-Ray Target or Magnet

Therac-25 Treatment Room



The Software Part of the Therac-25

- **What does the software actually do?**
  - Monitors the machine status
  - Accepts input about, and sets machine up for, the desired treatment
  - Turns the beam on in response to an operator command (assuming that certain operational checks on the status of the physical machine are satisfied)
  - Turns the beam off when treatment is completed, when an operator commands it, or when a malfunction is detected
- **Very little software documentation**
  - Almost no software specifications and no software test plan
  - Unit and software testing was minimal – focus on integrated system testing
- **Stand-alone, “real-time” operating system**
  - Not a commercial OS – written specifically for the Therac-25
  - Preemptive scheduler that handles critical and non-critical tasks
    - Critical tasks: treatment monitor, servo (beam/gun-control), housekeeper
  - Software allows concurrent access to shared memory
    - No synchronization apart from shared variables
    - Mutex (test and set operations) were not atomic
  - Race conditions played an important part in the accidents

Bad Assumptions!

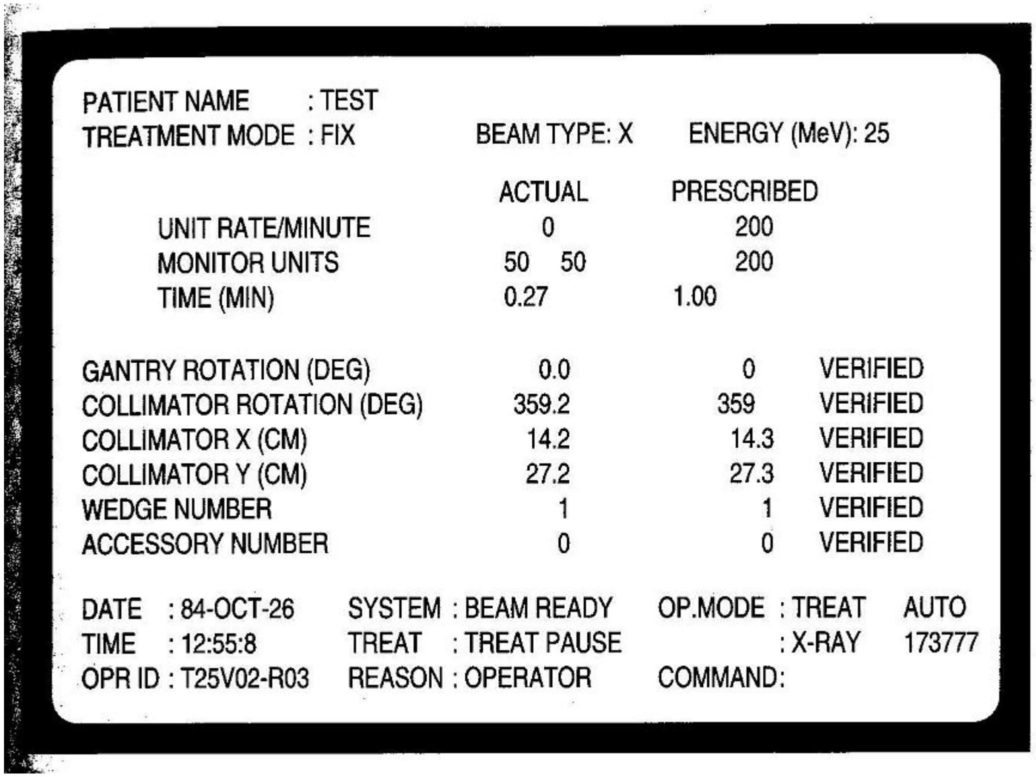
- **Manufacturer did a safety analysis**
  - Fault-tree analysis (What can go wrong? What can cause that?)
  - Excluded the software (assume software can’t cause a system fault)
  - No justification for numbers used in analysis – for example
    - Generic failure rate of 10^{-4} per hour for software events
    - *Prob*(Computer selects wrong energy) was assigned 10^{-11}
    - *Prob*(Computer selects wrong mode) was assigned 4 x 10^{-9}

- **Assumptions made in this analysis**
  1. Programming errors have been reduced by extensive testing on a hardware simulator and under field conditions on teletherapy units. Any residual software errors are not included in the analysis.
  2. Program software does not degrade due to wear, fatigue, or reproduction process.
  3. Computer execution errors are caused by faulty hardware components and by "soft" (random) errors induced by alpha particles and electromagnetic noise.

In other words – assumed software was perfect.

Operator Interface

- **Operator entered data manually and machine cross-verified**
- **Operators complained about data-entry process being time-consuming**
  - New "auto-complete" function to copy treatment data instead of operator data re-entry
- **Cryptic error messages**
  - "malfunction" + number (1-64, represents analog or digital channel number)
  - Operators became insensitive to malfunctions, never thought to affect patient safety



```
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:
Nancy Leveson

Selected Accidents

- **Kennestone, 1985: apparently 15,000 - 20,000 rads instead of 200 rads**
  - (500 rads is fatal if delivered to the whole body, but this was focused)
- **Ontario, 1985: 13,000 - 17,000 rads**
  - Identified turntable switches as a potential problem
  - If turntable is out of place, X-ray strength beam isn’t attenuated by target
- **Yakima Valley, 1985: overdose of unknown size**
  - Manufacturer argued it that an overdose was impossible; machine was safe
  - 1987: two more overdoses to additional patients
- **East Texas, 1986: 16,500 - 25,000 rads**
  - Operator initially type “X”-ray; but changed it to “e”lectron
  - Patient ended up pounding on door to be let out (intercom was broken)
  - Manufacturer again said overdose was impossible
  - **Second similar overdose three weeks later; patient died within 3 weeks**

What Went Wrong?

- **Overall, machine was known to be quirky**
  - Operators had to ignore or override error messages all the time to use machine
    - Gave false sense of security – “lots of safety shutdowns” instead of “why doesn’t this work?”
  - Errors poorly documented – difficult to know when accident had really occurred
  - No independent checks on correct operation; patient burns were only sign of problems

- **Initially blamed a position sensor sensitive to a single-bit error**
  - Table at boundary between shaft encoder values could give false readings
  - They didn’t use gray coding on shaft encoders!

There Were Software Defects

- **Tyler (East Texas) accident software problem:**
  - Changes ignored for an 8-second magnet moving window (even “x” to “e”)
    - So, changing from X-Ray to Electron might not change dose
    - But, X-Ray dose in Electron mode delivers 100x overdose, because diffuser is not in place to attenuate beam by 99%!
  - Race condition on whether prescription is completed or not
    - Changes made to operate inputs ignored during 8-second window
    - Screen showed changed version, but machine executed unchanged version
    - Problem with hand-off between user interface and dosing task

- **Yakima software problem:**
  - 8-bit rollover skips a safety check when passing through value zero
    - “0th” time in loop is an initialization … but re-executes every 256 iterations!
  - Operator hit “set” button precisely when rollover occurred
    - 1 in 256 chance of it happening
    - Unlikely to find this in testing, especially if you’re not looking for exactly this problem

Lessons Learned

- **Virtually all complex software can be made to behave in an unexpected fashion under certain conditions**

- **Safety analysis requires failure-rate numbers that are realistically derived**
  - Fault-tree analysis and other safety analysis methods are only as good as the numbers that you put into them for modeling

- **Fault-tolerance requires good backup mechanisms, with possible diversity, for safety-critical functionality**
  - Software mechanisms did not have a hardware interlock backup that might have provided a sanity check
  - If you don’t have a completely independent duplex system, you’re not going to be safe for a critical system

- **Race conditions can occur in time-critical software and you need to handle concurrency, shared variables and real time carefully**
  - This is 348/349 intro material

Could It Happen Again Anywhere?

- **2001 in Panama: Radiation Therapy Overdoses**
  - 28 patients affected
  - 5 deaths

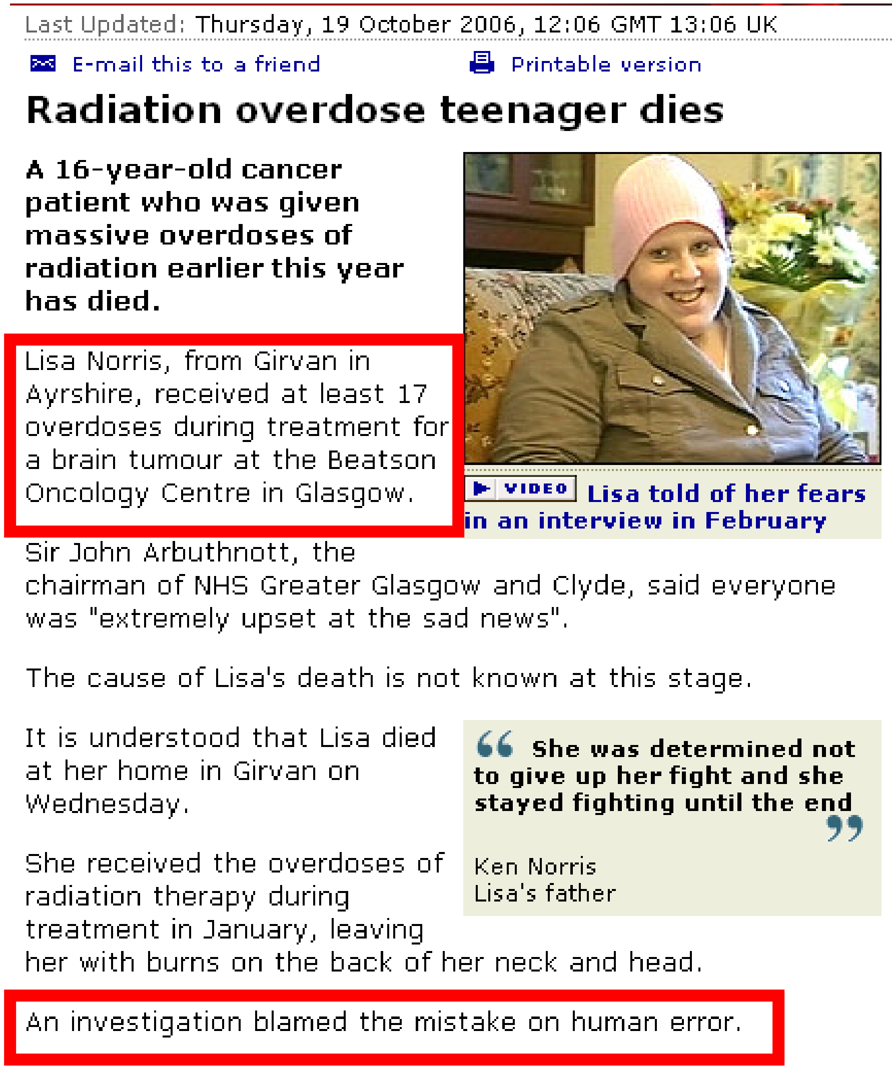
- **Due to planning software (i.e., how big a dose to use)**
  - “IAEA says technicians in Panama misused treatment planning software”
  - “patients received anywhere from 20%-100% more radiation than intended”
  - US spokesperson said it can’t happen here, because our treatments are given by a team of “highly trained and dedicated professionals”
  - http://www.aip.org/isns/reports/2001/020.html

Could It Happen Again In The US?

- **March 2005: Radiation therapy overdoses**
  - 77 patients affected in Tampa, Florida
  - 12 deaths (but unclear if due to overdoses or just due to cancer)
  - 50% overdose due to improper calibration when machine was installed
    - Second check by second physicist not performed as required
  - $1000 fine levied against hospital by Florida Bureau of Radiation Control
    - http://www.boston.com/yourlife/health/diseases/articles/2005/04/02/cancer_patients_exposed_to_high_radiation/

Could It Happen Yet Again Elsewhere?

- October 2006



Last Updated: Thursday, 19 October 2006, 12:06 GMT 13:06 UK

E-mail this to a friend   Printable version

Radiation overdose teenager dies

A 16-year-old cancer patient who was given massive overdoses of radiation earlier this year has died.

Lisa Norris, from Girvan in Ayrshire, received at least 17 overdoses during treatment for a brain tumour at the Beatson Oncology Centre in Glasgow.

▶ VIDEO Lisa told of her fears in an interview in February

Sir John Arbuthnott, the chairman of NHS Greater Glasgow and Clyde, said everyone was "extremely upset at the sad news".

The cause of Lisa's death is not known at this stage.

It is understood that Lisa died at her home in Girvan on Wednesday.

> **She was determined not to give up her fight and she stayed fighting until the end**
>
> Ken Norris
> Lisa's father

She received the overdoses of radiation therapy during treatment in January, leaving her with burns on the back of her neck and head.

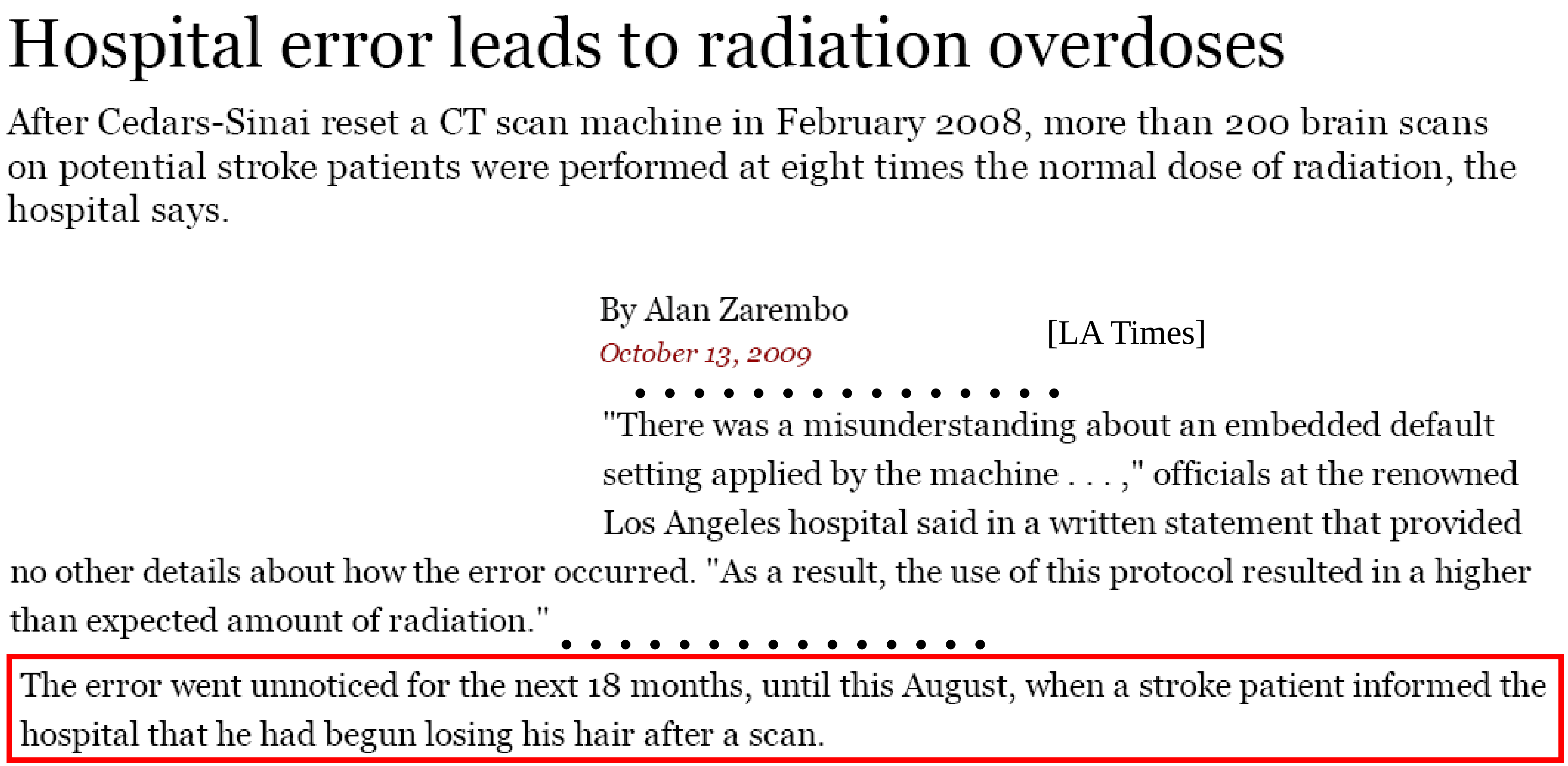
An investigation blamed the mistake on human error.

http://news.bbc.co.uk/1/hi/scotland/glasgow_and_west/6065040.stm

Paperwork error on treatment plan lead to the overdoses – given treatments for other patients
http://uk.news.yahoo.com/27102006/325/paperwork-error-led-radiation-overdose.html

Could It Happen *YET AGAIN* In The US?

- **October 2009: Brain Scan overdoses in Los Angeles**
  - 200 brain scans to look for stroke systems given 8x intended dose on CT system
  - New, tailored operation profile to give better data – but had a mistake
  - (Reports of broader CT scan overdoses being investigated a few years later)



Hospital error leads to radiation overdoses

After Cedars-Sinai reset a CT scan machine in February 2008, more than 200 brain scans on potential stroke patients were performed at eight times the normal dose of radiation, the hospital says.

By Alan Zarembo
October 13, 2009

[LA Times]

• • • • • • • • • • • • • • •

"There was a misunderstanding about an embedded default setting applied by the machine . . . ," officials at the renowned Los Angeles hospital said in a written statement that provided no other details about how the error occurred. "As a result, the use of this protocol resulted in a higher than expected amount of radiation."

• • • • • • • • • • • • • • •

The error went unnoticed for the next 18 months, until this August, when a stroke patient informed the hospital that he had begun losing his hair after a scan.

Additional Links

- **"An Investigation of the Therac-25 Accidents"**

  Nancy Leveson & Clark Turner

  http://courses.cs.vt.edu/~cs3604/lib/Therac_25/Therac_1.html

- **Therac-25 Case Materials**

  http://www.computingcases.org/case_materials/therac/therac_case_intro.html

- **What you should get out of this part of the lecture**
  - What the Therac 25 does
  - The general scenarios for the faults
    - What type of thing went wrong at the application level (general; not exact)
    - What underlying technical causes were (general; not exact)
    - Lessons Learned