Software Engineering

Computer Science Tripos 1B
Michaelmas 2011

Richard Clayton

Lecture Three

Critical software

- Many systems must avoid a certain class of failures with high assurance
- safety critical systems
  - failure could cause, death, injury or property damage
- security critical systems
  - failure could allow leakage of confidential data, fraud, ...
- real time systems
  - software must accomplish certain tasks on time
- Critical systems have much in common with critical mechanical systems (bridges, brakes, locks,...)
- Key: engineers study how things fail

Tacoma Narrows, Nov 7 1940

Definitions I

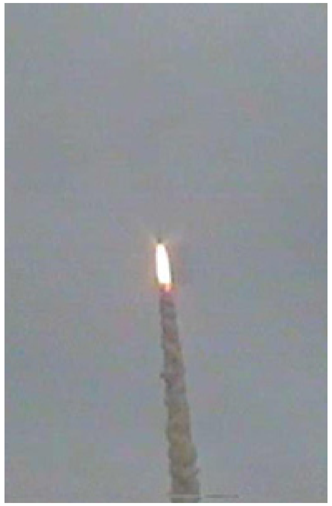
- Error
  - design flaw or deviation from intended state (a static quality)
- Failure
  - non-performance of system (a dynamic quality). Classical definition says “under specified environmental conditions”.
- Reliability
  - probability of failure within a set period of time
  - typically expressed as MTBF/MTTF: mean time between failures / to failure, depending whether system will be repaired and restarted
- Accident
  - undesired, unplanned event resulting in specified kind/level of loss
- Near Miss (or Incident)
  - event with the potential to be an accident, but no loss occurs

Definitions II

- Safety
  - freedom from accidents
- Hazard
  - set of conditions on system which in some environmental conditions, will lead to an accident
  - hence: hazard + failure = accident
- Risk
  - the probability of a bad outcome
  - the probability that hazard leads to accident (danger), combined with the hazard exposure or duration (latency)
- Uncertainty
  - risk not quantifiable

Arianne 5, June 4 1996






- Arianne 5 accelerated faster than Arianne 4
- This caused an operand error in float-to-integer conversion
- The backup inertial navigation set dumped core
- The core was interpreted by the live set as flight data
- Full nozzle deflection $\rightarrow$ 20^o angle of attack $\rightarrow$ booster separation
- $370 million of satellites destroyed

Real-time systems

- Many safety-critical systems are also real-time systems used in monitoring or control
- Criticality of timing makes many simple verification techniques inadequate
- Often, good design requires very extensive application domain expertise
- Exception handling tricky, as with Arianne
- Testing can also be really hard

Patriot missile failure, Feb 25 1991

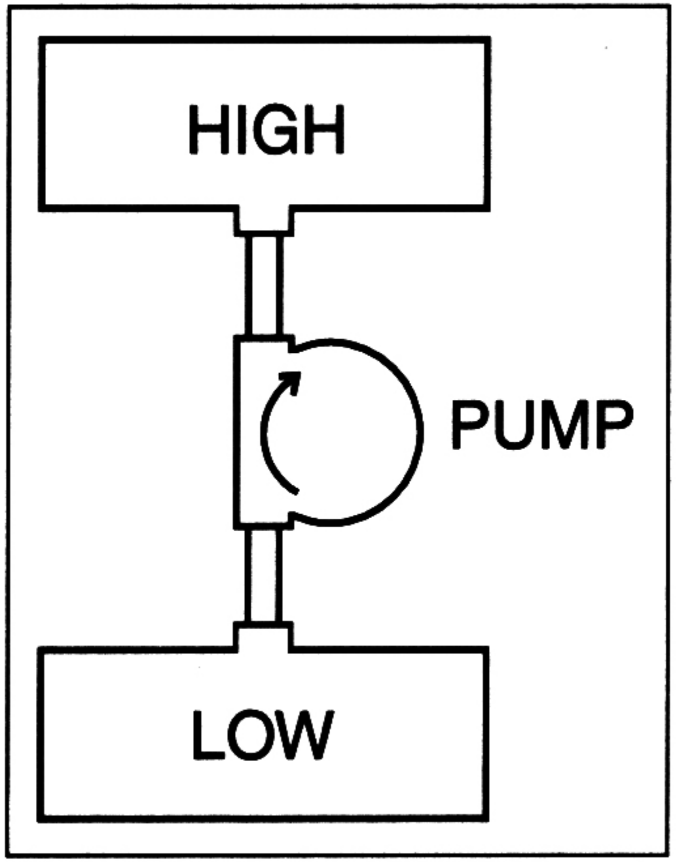
- Failed to intercept an Iraqi scud missile in First Gulf War
- SCUD struck US barracks in Dhahran; 29 dead
- Other SCUDs hit Saudi Arabia, Israel

Patriot missile failure

- Reason for failure
  - measured time in 1/10 sec, truncated from .0001100110011…
  - when system upgraded from air-defence to anti-ballistic-missile, accuracy increased
  - but not everywhere in the (assembly language) code!
  - modules got out of step by 1/3 sec after 100 hours operation
  - so system looked for Scud 600 metres away from where it was
  - since nothing visible at incorrect location, no launch occurred
  - not found in testing as spec only called for 4h tests
- Critical system failures are typically multifactorial: “a reliable system can’t fail in a simple way”
- But classical definition of `failure’ said “under specified environmental conditions”… So was this a failure?

Security critical systems

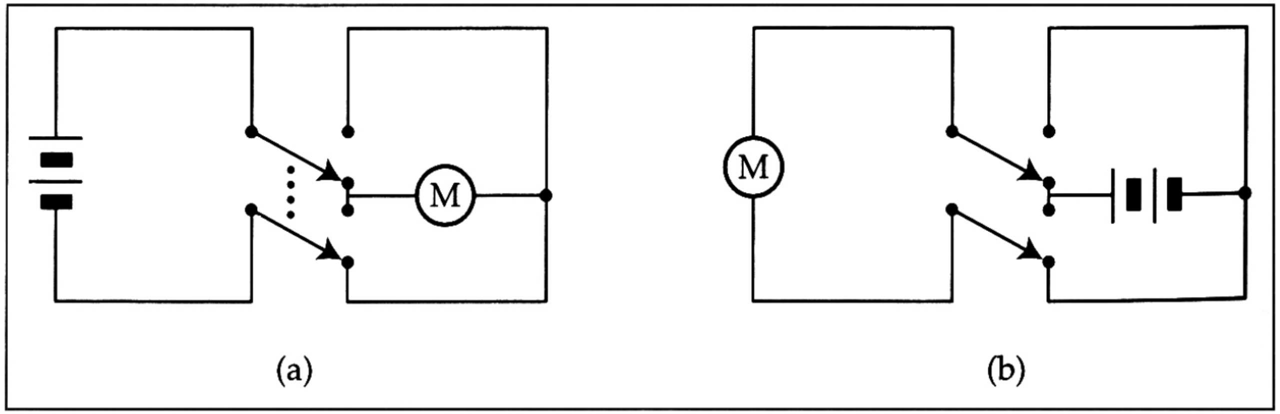
- Usual approach – try to get high assurance of one aspect of protection
- Example: stop classified data flowing from 'high' to 'low' using one-way flow
- Assurance via simple mechanism
- Keeping this small and verifiable is often harder than it looks at first!



Building critical systems

- Some things go wrong at the detail level and can only be dealt with there (e.g. integer scaling)
- However in general safety (along with security and real-time performance) is a system property and has to be dealt with at the system level
- A very common error is not getting the scope right
  - for example, designers don't consider human factors such as usability and training

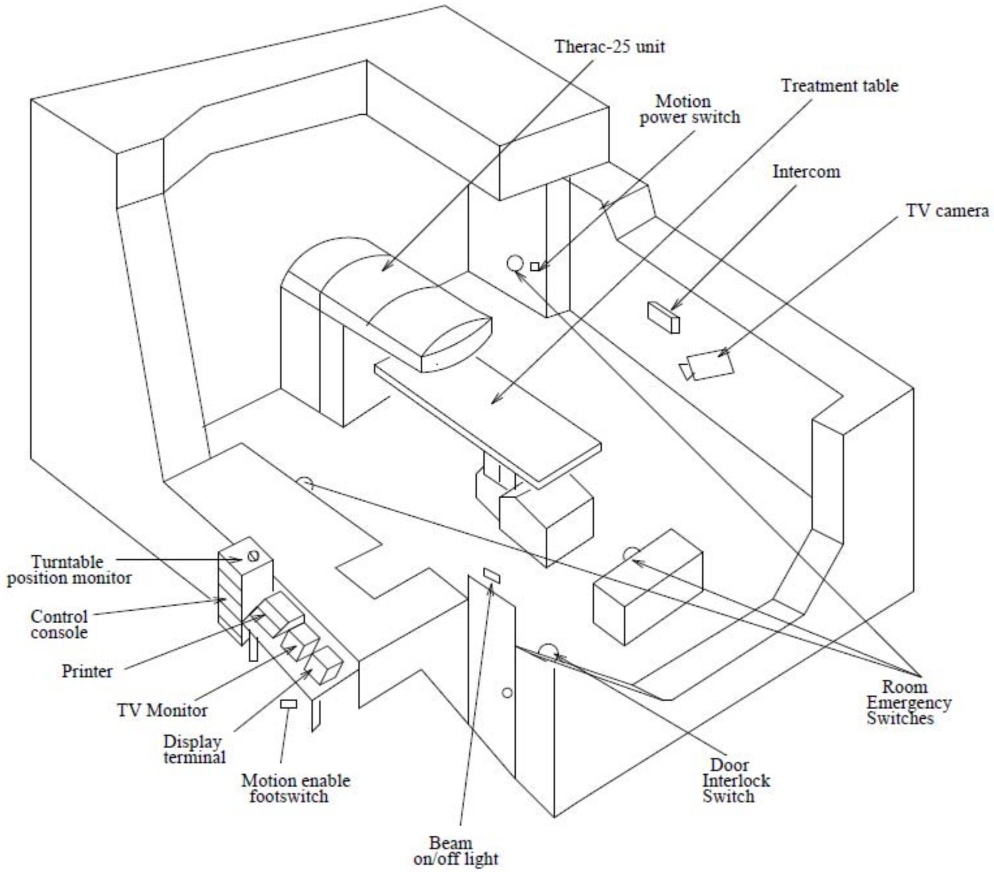
Hazard elimination



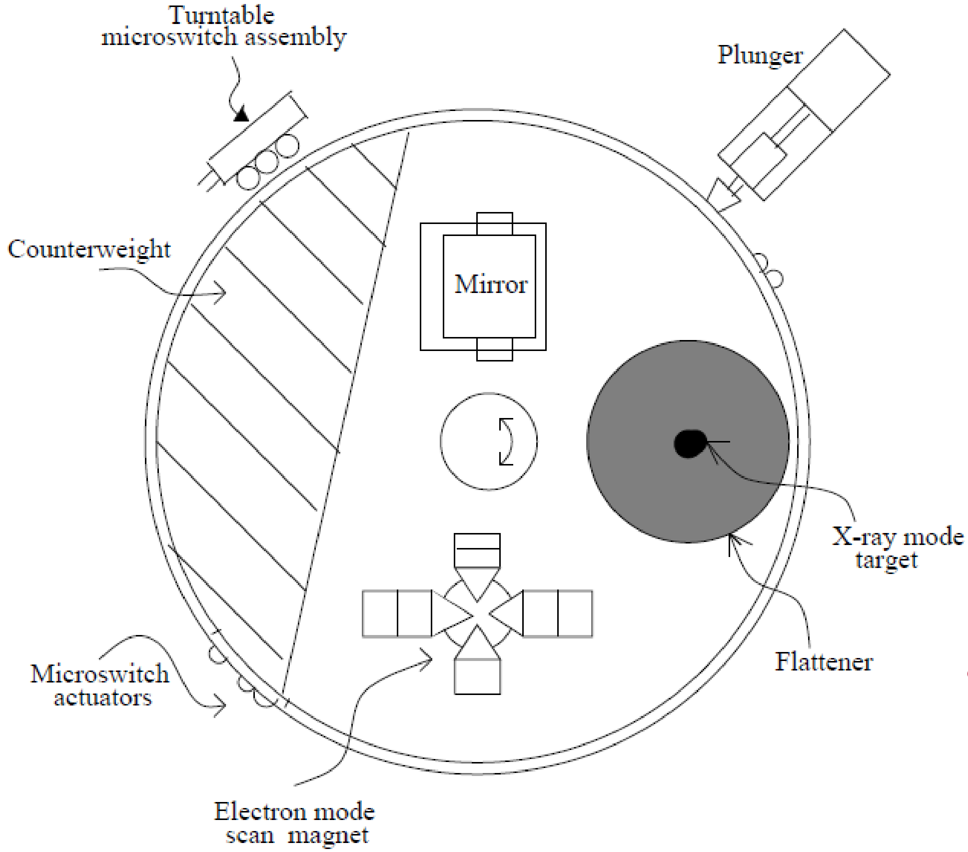
- e.g., motor reversing circuit above, in the left hand circuit failure of both switches to move together will short the battery
- Some tools can eliminate whole classes of software hazards, e.g. using a strongly-typed language such as Ada
- But usually hazards involve more than just software

The Therac accidents I

- The Therac-25 was a radiotherapy machine sold by AECL; 11 machines shipped
- Between 1985 and 1987 three people died in six accidents
- Example of a fatal coding error, compounded with usability problems and poor safety engineering



The Therac accidents II



- 25 MeV 'therapeutic accelerator' with two modes of operation:
  1. 25MeV focussed electron beam on target to generate X-rays
  2. 5-25 MeV spread electron beam for skin treatment (with 1% of beam current)
- Safety requirement
  - don't fire 100% beam at human!

The Therac accidents III

- Previous models (Therac 6 and 20) had mechanical interlocks to prevent high-intensity beam use unless X-ray target in place
- The Therac-25 replaced these with software
- Fault tree analysis arbitrarily assigned probability of 10^{-11} to ‘computer selects wrong energy’
- Code was poorly written, unstructured and not really documented

The Therac accidents IV

- Marietta, GA, June 85: woman's shoulder burnt.
  - settled out of court. FDA not told.
- Hamilton, Ontario, July 85: woman's hip burnt.
  - AECL suspected a micro-switch error (reporting incorrect turntable positions) but could not reproduce fault; changed software anyway.
- Yakima, WA, Dec 85: woman's hip burned
  - "could not be a malfunction"
- East Texas Cancer Centre, Mar 86: man burned in neck
  - died five months later of complications
  - 3 weeks later: another man burned on face & died after 3 weeks
- Hospital physicist managed to reproduce flaw:
  - if parameters changed too quickly from x-ray to electron beam, then the safety interlocks failed
- Yakima, WA, Jan 87: man burned in chest and died
  - different bug now thought to have caused Ontario accident

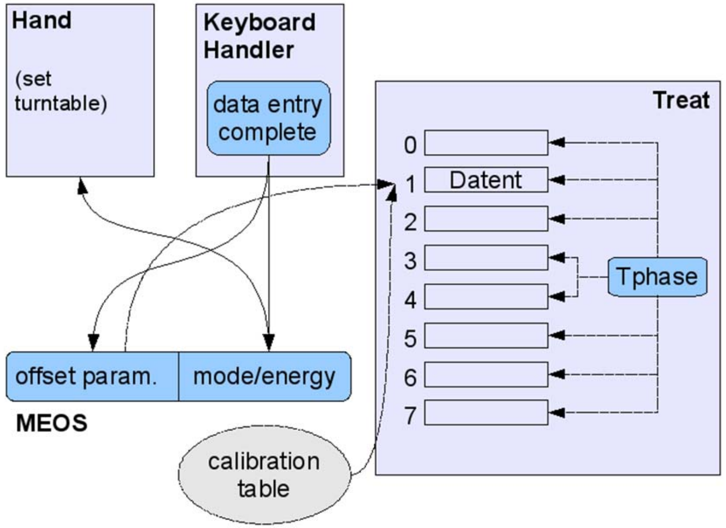
The Therac accidents V

```
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:
```

- East Texas deaths caused by editing ‘beam type’ and then issuing a start treatment request very quickly thereafter
- This was due to poor software design

The Therac accidents VI

- Data entry routine sets turntable and 'MEOS' (the mode and energy level)
- When data entry complete (cursor on last line) machine starts configuration



- Part of this involves setting magnets into correct position (which takes 8 seconds, so a timer routine is called)
- The timer routine checks for cursor movement if it is being called whilst the magnets are being moved
- Unfortunately, it also cleared the "magnets moving" flag; so it didn't check the cursor for subsequent magnet moves ☹

The Therac accidents VII

- AECL had ignored safety aspects of software
  - initial investigations had looked for hardware faults
- Confused reliability with safety
- Lack of defensive design
- Inadequate reporting, follow-up and regulation – failed to explain Ontario accident at the time
  - a true/false flag was being incremented to keep it true, and after 255 increments it speciously got set to the wrong value!
- Unrealistic risk assessments (‘think of a number and double it’)
- Inadequate software engineering practices
  - specification an afterthought, complex architecture, dangerous coding, little testing, careless HCI design, incomprehensible messages displayed to users, failure to follow up accident reports