Engineering High-Dependability Systems (2)

CS3 / SEOC1

Note 17

The Safety Case

A safety case is a document, or collection of documents, that presents the arguments for believing that a proposed potentially-dangerous system is acceptably safe. It sets out the risks involved with the operation of the process or equipment and the possible consequences of failure, and specifies what will be done (or has been done) to minimise the probability and the impact of these failures.

Design for assessment: Arguments of software safety

process versus product:
design process used will lead to safe system **versus** system produced is shown to be safe.

qualitative versus quantitative:
claims of *good quality* (e.g. unsafe states not possible) **versus** numerical *measures* of, for example, probability of error

formal verification:
use of formal methods. However, formal models often constructed but not verified

adoption of traditional engineering analyses:
e.g. HAZOPS (HAZard and OPerability Studies), FMEA (Failure Modes and Effects Analysis), FTA (Fault Tree Analysis)

software specific standards:
e.g. MOD 00-55, IEC 61508

responsibility:
open issue – responsibility versus liability

Safety Standards for Software (1)

MOD Def.Stan 00-55 *Procurement of Safety Critical Software in Defense Equipment*

- Requirements:
  - safety management
  - software engineering practice
- Guidance:
  - lifecycles
  - hazard analysis
  - risk analysis
  - V & V
  - independent auditing
  - ...

Safety Standards for Software (2)

IEC 61508 *Functional Safety of electrical/ electronic/ programmable electronic safety related systems*

- Safety at the system level
  - The overall safety lifecycle
  - Management of functional safety
  - Hazard analysis, risk reduction and safety integrity levels
- Hardware and architectural safety
  - Managing hardware
  - Failure probabilities
  - Integrating hardware into the safety lifecycle
- Software safety
  - Safety related software
  - Selecting development methods and tools
  - Verification and validation

Computer-related accidental death: An empirical exploration *D. MacKenzie, 1994*

- **Computer-related** – as before
- **accidental death** – non-deliberate (ie, military); empirically "easy" to measure

- Causes:

  4% physical (chiefly electromagnetic interference)

  3% software error:
  Therac-25 (2) + Patriot (28)

  92% failure in human-computer interaction
- Estimated number of deaths (until end 1992): *1,100 +/- 1,000*

Fatalities due to software error

- 1986, Therac-25:
  - radiotherapy machine
  - two operating modes:
    - low intensity, wide spread
    - high intensity, tight focus
  - software error in data entry permitted high intensity, wide spread
  - hardware interlock in Therac-20 removed
  - overdosing leads to 2 confirmed deaths
- 1991, Patriot anti-Scud missile system:
  - internal clock uses tenths of seconds
  - binary rounding error known
  - long run times not anticipated
  - 25th February: Alpha Battery in uninterrupted operation for over 100 hours; unable to track Scud missiles; 28 US servicemen killed
  - 26th February: software fix arrives

Fatalities due to human-computer interaction failures (1)

- 1992, A320 airbus crashes after over-rapid descent, 87 die. "Glass cockpit" descent display of 3.3 (thousands of feet per minute) wrongly interpreted as angle – Airbus denies responsibility (although interface changed subsequently).
- Similar incidents:
  - Habsheim 1988, Bangalore 1990, Moscow 1991, Nagoya 1994, Toulouse 1994, Paris 1994, ...
- 1988, USS *Vincennes* shoots down Iran Air airliner, all 290 on board die: weapon system human interface deemed "not optimal"

Lufthansa Airbus A320 Crash, Warsaw

- 14 Sept. 1993, flight DLH 2904 from Frankfurt overruns runway on landing at Warsaw Airport, Poland; co-pilot and 1 passenger die, 54 people hospitalised
- autopilot found to have prevented reverse thrust braking for 9 seconds
- specification of checks too severe (height, wheel-spin, weight on *both* wheels)
- dangers of in-flight reverse thrust activation: May 1991, Boeing 767 brought down over Thailand, killing all on board
- Pilot error/specification error? – Airbus denies responsibility (although code subsequently changed)

Fatalities due to human-computer interaction failures (2)

- 1992, London Ambulance Service Computer Aided Despatch System: primarily management failure, claims of 20-30 deaths
- 1982-1991, North Staffs. Royal Infirmary radiotherapy: double error-correction leads to underdosing of around 1,000 patients, 401 die – clinical verdict is "tens rather than hundreds" due to double error-correction
- 1994, Toulouse: Airbus A330 stalls during testing. Flight level set at 2,000ft instead of 7,000ft. 7 dead including Airbus chief test pilot
- attitudes to "safety envelope" different from Airbus and Boeing

Where is the danger?

- software does not directly cause fatalities
- interaction between software and physical world
- all software errors latent; present in specification:
  - requirements missed, incorrect or misunderstood
  - failure to correctly implement specification rarer
  - eg, NASA Galileo/Voyager mission critical software

How dangerous is software?

- Estimated number of deaths (until end 1992): *1,100 +/- 1,000*
- Compare with UK road deaths for 1992 alone: *4,274*
- human *perception* of risk and danger complicated:
  - degree of personal control
  - anticipated benefit
  - availability of data
- *Hypothesis:* software is safe, because we believe it to be dangerous

Where does SEOC fit in? Software engineering *versus* physical engineering

- art – craft – science ???

  science: strong theoretical foundation, *knowledge-based* teaching

  craft: little theory, *skills-based* apprentices

  art: highly subjective, *innate ability* crucial

- errors are *latent* (often in requirements or specification)
- physical systems *convex*:
  - e.g. beam is tested for 100kg load and 1000kg load, can assume will carry any load *between* 100-1000kg
- software not convex, but *structured*
  - can *partition* system; e.g. independence of failures in interface and operation (c.f. Therac 25)

SEOC and Dependability: Process, assessment and components

process oriented assessment:

- e.g. lifecycle models, project planning and management, structured test plans, ...
- combining qualitative and quantitative modelling and assessment

avoiding failures in communication:

- during design (c.f. Mars Climate Orbiter)
- during assessment (c.f. Voyager and Galileo; Arianne 5)
- validation as well as verification (c.f. LASCAD)

(reuse of) trusted/tested components: e.g. 3rd party developrs of high-dependability components

principled composition of components: theory of composition, e.g. independence of failures in components