Assessing the probability of failure of 1-out-of-2 software-based systems: now you *can* multiply two small numbers …

Bev Littlewood

Centre for Software Reliability, City University London

with

John Rushby (SRI, Menlo Park)

Andrey Povyakalo (CSR, City University London)

[Work mainly funded in C&I Nuclear Industry Forum and EPSRC projects]

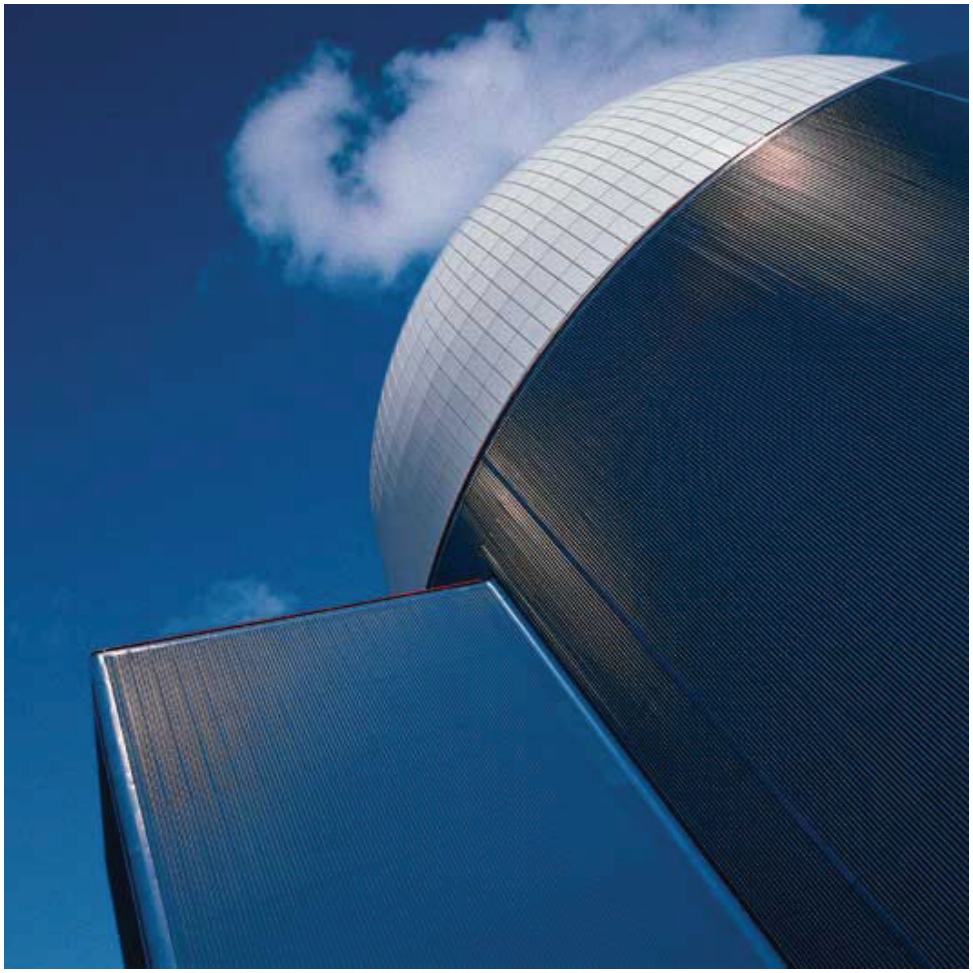
Background - a little history of a couple of old technical controversies concerning fault tolerance, and a new one

Do you remember 10^{-9} and all that?

25 years ago: much controversy about apparent need for 10^{-9} probability of failure per hour for flight control software

- Could it be achieved? Could such a claim be justified? *With what confidence?*

Or UK Sizewell B nuclear plant?



Protection system required 10^{-7} probability of failure on demand

Fault tolerance

- These are both examples of fault tolerant systems
- In the case of the A320, 2 *diverse software* channels
  - Rather complex architecture, with reversion to successively less functional modes
  - Licensing process did not allow any benefit to be claimed for the fault tolerance: largely because of doubts about independence of failures of different software versions
  - The required *figure*, 10^{-9}, was not directly addressed
- In the case of Sizewell, one software-based channel (PPS) and the other hardware only (SPS)
  - Controversy centred on PPS: how good was it? *Numeric* requirement
  - Initially required 10^{-4} *pfd* for PPS, 10^{-3} *pfd* for SPS
  - Eventually claimed 10^{-3} for PPS, 10^{-4} for SPS
  - Multiplying these to get 10^{-7} did not seem to be a problem….

How did these turn out?

- Sizewell B was licensed for operation; no software failures have been reported in operation
  - licensing was very costly, in spite of modest goal
- A320 family very successful, and *eventually* has demonstrated a low accident rate
  - several accidents in early service
  - Airbus claim none of these attributable *directly* to software
- Highly computerised current generation of aircraft seem safer than previous generations [see Boeing statsum]
  - E.g. A320/321/319/318 has hull loss rate of 0.34 per million departures
  - Boeing 777 may be even better
- But this is after-the-fact judgment: could it have been assured before massive operational exposure?

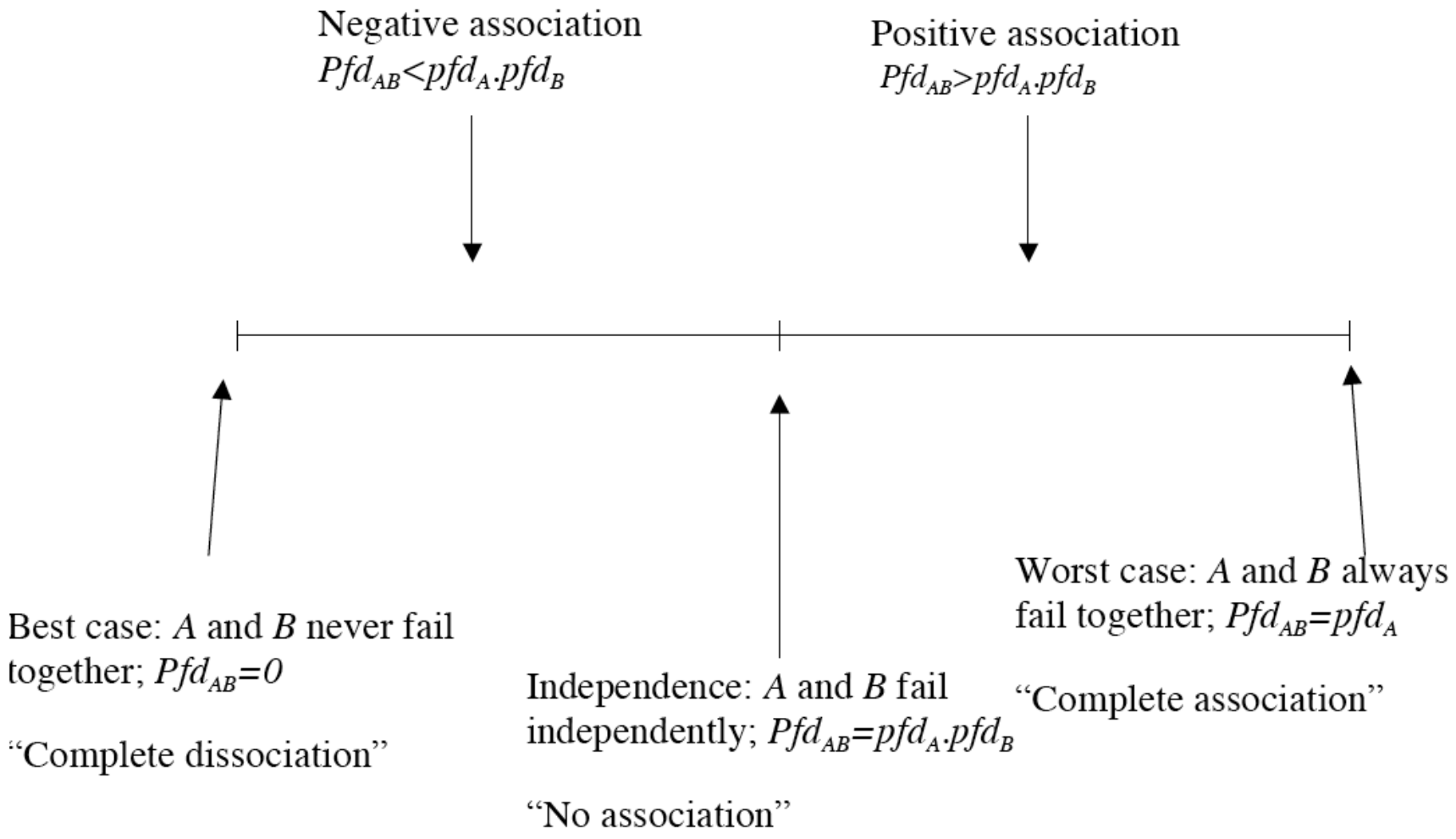
More recently: EdF/Areva's protection system for proposed UK EPR

- Goal is 10^{-9} probability of failure on demand (*pfd*) for the system
- Originally proposed 1-out-of-2 system
- Claim 10^{-4} *pfd* for one channel, 10^{-5} *pfd* for the other and then multiply these two numbers for system *pfd*
- Objections from Office of Nuclear Regulation (ONR)
- Currently, proposed to add a third channel (non-computer-based) and claim 10^{-2}, 10^{-4}, 10^{-3} respectively for the three channels
- …and multiply these numbers…?
- See HSE website for extensive, fascinating, documentation

Does software-diverse fault tolerance *work?* What is the evidence?

- Some successful industrial experience
  - E.g. Airbus, railway signalling, nuclear (e.g. Temelin)
- Extensive experiments, e.g. by Knight and Leveson and others
  - E.g. 2-out-of-3 systems better than single versions *on average*
  - But large variation in efficacy (e.g. best 'singles' better than worst 'triple')
  - AND strong evidence that versions do *not* fail independently
  - So system reliability depends on version reliabilities *and on dependence between them*
- Modelling work explains the non-independence
  - "difficulty" variation

Spectrum of dependence (1-o-o-2 case)



Can't assume independence

In Act 2 of Gilbert and Sullivan's *The Yeoman of the Guard*, Fairfax asks: "And didst thou see all this?"

Point replies: "Aye, with both eyes at once – this and that. The testimony of one eye is naught – he may lie. But when it is corroborated by the other, it is good evidence that none may gainsay. Here are both present in court, ready to swear to him."

Quoted by William Kruskal, "Miracles and Statistics: The Casual Assumption of Independence", *Journal of the American Statistical Association*, vol 83, Dec 1988, pp 929-940. http:/www.jstor.org/stable/2290117 (it's a good read!)

There's good news, *and bad news*…

- Pretty clear evidence for the efficacy of the fault tolerant approach in *achieving* reliability, in some *average* sense: design diversity is A Good Thing
  - But variation in what might be expected *in particular*
  - Cannot assume that the average efficacy will apply to a particular system
- Serious problems in assessing what has actually been achieved in a particular case
  - Claims of independence cannot be supported with certainty
  - Need to know *how dependent* the version failure processes are

So…an impasse? Maybe not

Our recent work has addressed problems of quantitative (probabilistic) assessment of software-diverse fault tolerant systems (usually 1-out-of-2 systems).

- Major problems concern (lack of) independence at aleatory and epistemic levels

We have some results that "solve" these problems

- but generally at the price of (possibly considerable) *conservatism*
- and there *remain* unsolved problems (some of which look *hard*)

Problem 1: Aleatory dependence

Consider a 1-out-of-2 "on demand" system, such as a protection system. We want system's probability of failure on demand:

- cannot assume that $pfd_{sys}=pfd_A.pfd_B$
- i.e. cannot assume independence of failures, so need to know "how dependent" the failures of the different channels are
- *Measuring* this dependence turns out to be as hard as measuring pfd_{sys} by treating the system as a black box
  - And will be infeasible when the system requirement is very stringent

Solution 1: LR model

[For details see Littlewood, B. & Rushby, J. (2012) – list of references at end of these slides.]

Consider a 1oo2 system in which channel *A* is "highly functional", and therefore complex, *but channel* B *is simpler and thus possibly "perfect"*

- For *A* our uncertainty concerns whether it will fail on a randomly selected demand: probability pfd_A
- For *B* our uncertainty concerns whether it is not perfect: probability pnp_B

Perfect software

- This property cannot be about *some* executions of the software
  - Like how many fail
- Must be a property about *all* executions, like correctness
- But correctness is relative to specifications, which themselves may be flawed
- We want correctness relative to the critical claims
  - Taken directly from the system's assurance case
- Call that perfection
- Software that will never experience a failure in operation, no matter how much operational exposure it has

Possibly perfect software

- You might not be certain a given piece of software is perfect
- But you might concede it has a possibility of being perfect
- And the more V&V it has had, the greater that possibility
- So we can speak of a (subjective) probability of perfection
- For a frequentist interpretation: think of all the software that might have been developed by comparable engineering processes to solve the same design problem as the software at hand
  - And that has had the same V&V, etc
- The probability of perfection is then the probability that any software randomly selected from this class is perfect
  - "Developing a particular program" is this "random selection"

Aleatory uncertainty for 1-o-o-2 system

Assume (for now) pfd_A and pnp_B are known. It is conservative to assume that if B is imperfect it fails whenever A does:

$$P(\text{system fails on randomly selected demand} \mid pfd_A, pnp_B)$$

$$\leq P(A \text{ fails}, B \text{ not perfect} \mid pfd_A, pnp_B)$$

$$= P(A \text{ fails} \mid B \text{ not perfect}, pfd_A, pnp_B)$$

$$\times P(B \text{ not perfect} \mid pfd_A, pnp_B)$$

$$= P(A \text{ fails} \mid pfd_A, pnp_B) \times P(B \text{ not perfect} \mid pfd_A, pnp_B)$$

$$= pfd_A \times pnp_B$$

B's imperfection tells us nothing about whether *A* will fail on *this* demand

Aleatory uncertainty (contd.)

So we have

P(system fails on randomly selected demand | pfd_A, pnp_B)

≤ $pfd_A . pnp_B$

Which contrasts with what we must conservatively assume for two certainly fallible channels:

P(system fails on randomly selected demand | pfd_A, pfd_B)

≥ $pfd_A . pfd_B$

- The events "*A* fails" and "*B* not perfect" are (conditionally) independent at the aleatory level
- This is not true of "*A* fails" and "*B* fails"

This "solves" the problem of aleatory dependence

…i.e. we *can* now multiply together two small numbers to get a very small number for system *pfd*, albeit at the price of conservatism

- If we know pfd_A and pnp_B we can simply multiply them to get a conservative value for pfd_{sys}

At this point I should perhaps declare victory and end this presentation. But…

Problem 2: Epistemic dependence

- We don't know values of pfd_A and pnp_B with certainty
- We can represent this *epistemic uncertainty* formally by

  $F(p_A,p_B)=\Pr(pfd_A<p_A,pnp_B<p_B)$

  – Can think of this as an assessor's Bayesian posterior distribution if he has evidence from testing, verification, other kinds of analysis (e.g. analysis of complexity of problem/solution), etc, etc
- If we know this, we can, for example, get the unconditional (subjective) conservative (upper) bound on probability of system failure:

$$\int p_A p_B \, dF(p_A, p_B)$$

- But can assessor tell us what his F is?

Epistemic dependence problem (contd)

- Most assessors would find it hard to tell us what their F is
- In particular, it is *dependence* between beliefs about pfd_A and pnp_B that is the problem
- Assessors may be able express *marginal* beliefs about the parameters separately

Solution 2: Epistemic dependence

[See Littlewood, B. & Povyakalo, A. A. (2013) for details]

- Here we have developed a conservative approach that involves only *marginal beliefs* about pfd_A and pnp_B
- *We do not need to know about the dependence between these beliefs*
- We have several results, based upon differently-expressed beliefs that an assessor might reasonably have about pfd_A and pnp_B. For example…..

Solution 2 (contd)

- For example, *confidence* bound: we show that if

$$P(pfd_A < p_A) = 1 - \alpha_A$$

$$P(pnp_B < p_B) = 1 - \alpha_B$$

then

$$P(pfd_{sys} < p_A \times p_B) > 1 - (\alpha_A + \alpha_B)$$

  - i.e. *multiply* the claims, *add* the doubts
  - E.g. 99% confident $pfd_A < 10^{-4}$, 98% confident $pnp_B < 10^{-2}$, then 97% confident that system pfd better than 10^{-6}

- That is, we have a confidence bound for system *pfd* based only on *marginal beliefs* about the parameters
- We have other results, e.g. bounds on $E(pfd_{sys})$ – see our paper

What has all this achieved?

- Littlewood/Rushby (LR) "solves" problem of dependence at the aleatory level (i.e. dependence between failures of *A* and *B*)
  - At the price of conservatism
- Littlewood/Povyakalo (LP) "solves" problem of dependence at the epistemic level (i.e. dependence between an assessor's beliefs about parameters pfd_A and pnp_B)
  - At the price of further conservatism
- We have reduced the problem to one of expressing marginal beliefs about both pfd_A and pnp_B

At this point I should DEFINITELY declare victory, and end this presentation. But….

Problem 3: what is *pnp*?

- So there just remain problems concerning marginal beliefs about the parameters pfd_A and pnp_B
- The first of these is easier – for example can use statistical analysis from operational testing (e.g. see Littlewood and Wright, 2007)
- That leaves pnp_B, which is more problematic: it is the subject of our current work

What evidence supports claims for *pnp*?

There seem to be four main candidates:

- Evidence from testing
  - Extensive *failure-free* testing (or real-life operation)
- Evidence of quality of development process
  - In particular, previous successful (e.g. non-failing) products
- Evidence from formal verification
- "The problem is utterly simple; our solution is also utterly simple – so why would it not be perfect?"
  - Such arguments can be made less brashly, and can be very compelling…

The list here is roughly in order of the ease of using the evidence *to make claims in probabilistic form*

- But apparently not same order as "*strength*" of evidence

This is 'work in progress'…

The problem looks surprisingly hard.

- Evidence from failure-free working during test *does* support probabilistic perfection claims, but apparently only weakly
  - It's hard to discriminate between "perfection" and "very small *pfd*"
- In principle we can model *process evidence* probabilistically
  - E.g. Siemens (and others) have built previous protection systems that have been "successful" (e.g. have not failed, so *may* be perfect)
  - This may also be weak evidence in practice
    - Seems unlikely that there are *many* past systems
    - We would not *know* they were perfect

Other evidence (contd)

- Intuitively, evidence for "utter simplicity" (of problem and solution) seems the most compelling of all
  - At least to me!
  - But how do you turn this into statements of (probabilistic) confidence in perfection?
- Ditto, verification evidence
  - Uncertainty arises from fallibility of proof, incompleteness of formal specification, etc
  - This *can* be modelled (for example, we did it for *pfd* rather than *pnp* in Littlewood and Wright (2007)), but how do you populate the model with numbers?
- In practice, *all* these disparate kinds of evidence need to be combined to support *pnp* claims
  - BBNs?

Comment on the *pnp* idea

- "Probability of perfection" provides a bridge between correctness-based verification activities and probabilistic claims needed at the system level
- Relieves formal verification, and its tools, of the burden of infallibility
  - And such claims of infallibility were never believable, were they…?

A few concluding comments

Of course all this is no magic solution. On the positive side:

- The handling of aleatory uncertainty is greatly simplified compared with the case of two *certainly fallible* channels
- So is the problem of epistemic uncertainty
  - Only two parameters, compared with three
  - No need for beliefs about dependence

On the other hand

- Inference about perfection not easy
- The results are conservative
- The architecture *is* a special one
  - but it is very plausible for certain applications
  - E.g. as a means of *achieving* reliability for, say, a protection system; or for functional channel plus monitor; or highly functional channel plus get-you-home channel

Thank you for listening!

Questions? Brickbats…?!

References

Littlewood, B. & Rushby, J. (2012). "Reasoning about the Reliability of Diverse Two-Channel Systems in which One Channel is 'Possibly Perfect'", *IEEE Trans Software Engineering*, vol 38, no 5, pp 1178-1194, 2012. Free download: http://openaccess.city.ac.uk/id/eprint/1069

Littlewood, B. & Povyakalo, A. A. (2013). "Conservative reasoning about epistemic uncertainty for the probability of failure on demand of a 1-out-of-2 software-based system in which one channel is 'possibly perfect'" Accepted for publication in *IEEE Trans Software Engineering*. Free download: http://openaccess.city.ac.uk/id/eprint/1611

References (contd.)

Littlewood, B. and D. Wright (2007). "The use of multi-legged arguments to increase confidence in safety claims for software-based systems: a study based on a BBN of an idealised example" *IEEE Trans Software Engineering* vol 33, no 5, pp 347-365, 2007 http://openaccess.city.ac.uk/1619/

Boeing Statsum, "Statistical Summary of Commercial Jet Airplane Accidents, Worldwide Operations, 1959-2011", Boeing Commercial Airplanes, July 2012. Available for free download: www.boeing.com/news/techissues/pdf/statsum.pdf

CITY UNIVERSITY
LONDON

CSR Building confidence in a computerised world

www.csr.city.ac.uk