Limits to Dependability Assurance - A Controversy Revisited (Or: A Question of 'Confidence')

Bev Littlewood

Centre for Software Reliability, City University, London

b.littlewood@csr.city.ac.uk

[Work reported here supported by UK Engineering and Physical Sciences Research Council under DIRC and INDEED projects]

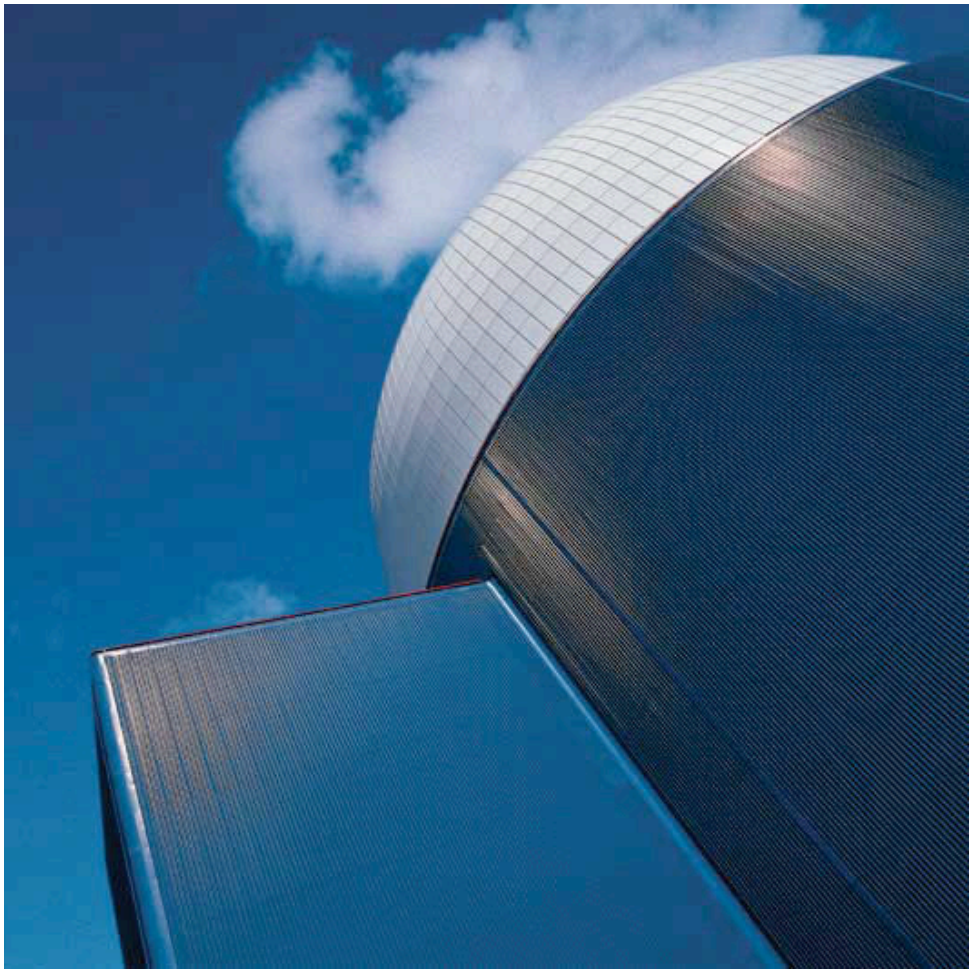
Background, a little history of a couple of technical controversies…

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

Twenty 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?

Or UK Sizewell B nuclear plant?



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

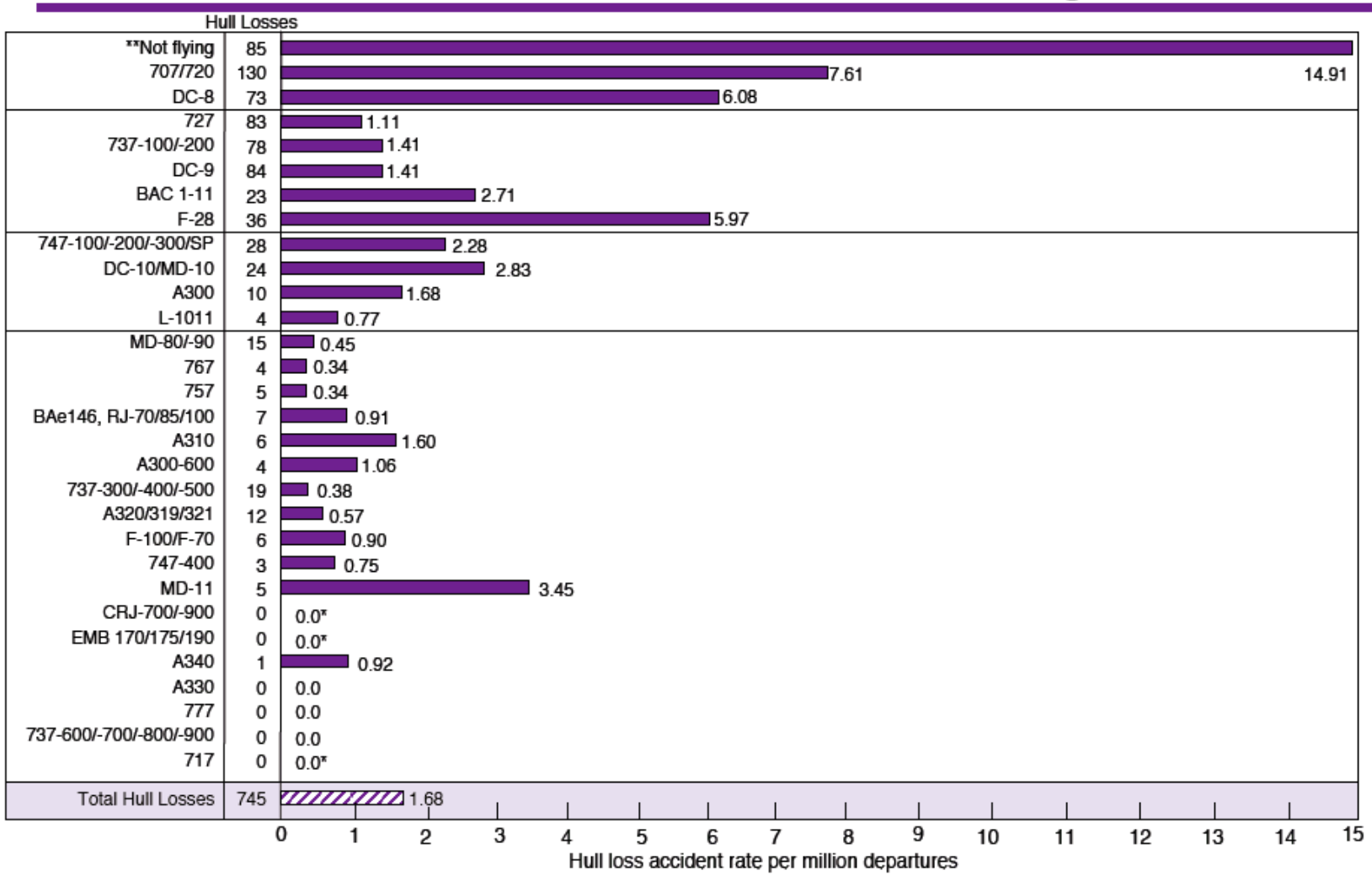
- Diversity: software-based primary system (PPS), hardwired secondary system (SPS)
- Controversy centred on PPS: how good was it?
  - initially required 10^{-4} for PPS, 10^{-3} for SPS
  - eventually claimed 10^{-3} for PPS, 10^{-4} for SPS

How did these turn out?

- Sizewell B 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
- There are interesting statistics on accident rates of current generation of 'computerised' aircraft

Accident Rates by Airplane Type

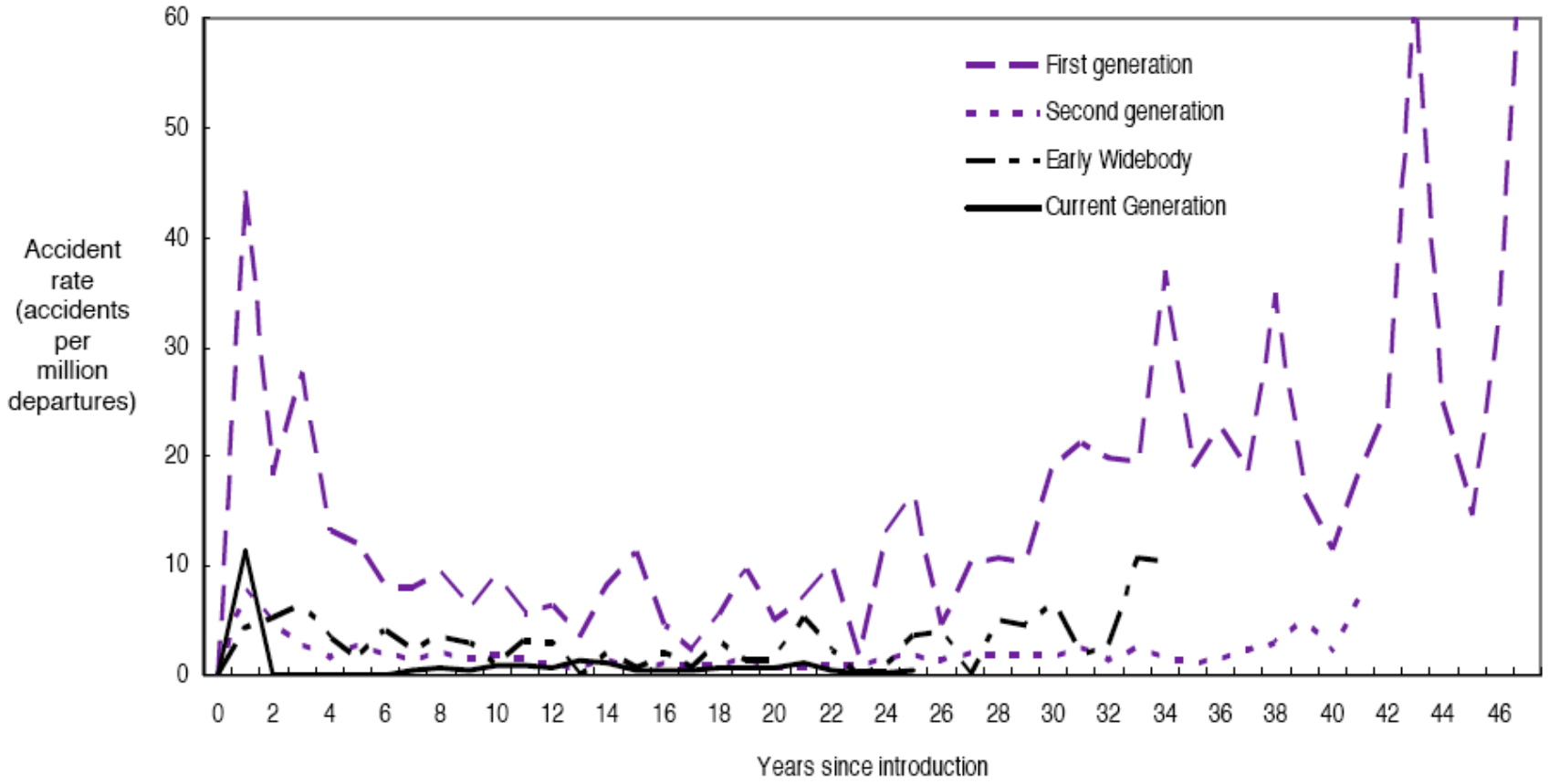
Hull Loss Accidents – Worldwide Commercial Jet Fleet – 1959 through 2005



Source: "Statistical summary of commercial jet airplane accidents", Boeing Commercial Airplanes

Accident Rates by Years Following Introduction

Hull Loss and/or Fatal Accidents – Worldwide Commercial Jet Fleet – 1959 through 2005



Source: "Statistical summary of commercial jet airplane accidents", Boeing Commercial Airplanes

What does this tell us?

- Highly computerised current generation of aircraft seem safer than previous generations
  - Those types having large fleets seem very safe
- But there are significant differences between aircraft types
  - E.g. B737 family seems better than A320 family
  - E.g. B777 record is very good
- Early life losses - from *some* of the aircraft types - contribute disproportionately to the accident rates
- But this is after-the-fact judgment: could it have been predicted before operation?
- In particular, could the contribution of computer-based systems have been *predicted*?

The nature of the problem

Why can't software be fault-free?

Difficulty, complexity, novel functionality…. all militate against perfection: software *will* contain faults

What are achieved fault densities?

- Even for safety-critical industries, 1 fault per kLoC is regarded as first class
  - e.g. study of C130J software by UK MoD estimated 1.4 *safety-critical faults* per kLoC (23 per kLoC for non-critical)
- For commercial software, studies show around 30 faults per kLoC
  - Windows XP has 35 MLoC, so >1 million faults?!
- But this does *not* necessarily mean software must be unreliable…

Many faults = very unreliable?

Not necessarily!

- Microsoft Windows reliability has grown from 300 hours MTBF (with 95/98) to about 3000 hours *despite increased size and complexity* (i.e. more faults)
- *After-the-fact estimation* of failure rates, based on extensive operational experience with software in aircraft and automobiles suggest very high reliabilities *can* be achieved
  - Automobiles: Ellims has estimated that no more than 5 deaths per year (and about 300 injuries) caused by software in the UK - suggests about 0.2×10^{-6} death/injury failures per hour. Even better *per system* - say 10^{-7}
  - Aircraft: *very* few accidents have been attributed to software; Shooman claims, again, about 10^{-7} per hour per system
  - But these *are* after-the-fact figures

Why can software be so reliable…

…when it contains thousands of faults?

- Because many (most?) faults are ‘very small’
  - i.e. they occur extremely infrequently during operation
- Adams - more than twenty years ago - examined occurrence rates of faults on large IBM system software: found that more than 60% were ‘5000-year’ bugs
  - i.e. each such bug only showed itself, on average, every 5000 years (across a world-wide population of many users)
  - figures based on *reported* bugs - may be even more dramatic if *unreported* ones could be included?
  - so the systems he studied had many thousands of these faults, but were acceptably reliable in operation

So what's the problem?

- Just because large complex programs *can* be very reliable, it does not mean you can assume that a particular one *will* be
  - even if you have successfully produced reliable software in the past, you can't assume from this that a *new* program will be reliable
  - even if some software engineering processes have been successful in the past, this does not *guarantee* they will produce reliable software next time
- So you need to measure how reliable your software *actually is*
- And this assessment needs to be carried out *before* extensive real-life operational use
  - how else can you make a risk assessment?

So what's the problem?

We need to be able to tell, *before extensive operational experience is available*, that a system is good enough

- E.g for critical aircraft systems, 10^{-9} probability of failure per hour
  - This is not as silly as it seems: if we want 10^{-7} for the whole aircraft - and this is being achieved - and there are ~100 such systems per aircraft, then that is ~10^{-9} per system
- This is extremely difficult to achieve, it seems even harder - some would say impossible - to assure
- Even for the Sizewell PPS - with a very modest dependability goal - it proved very difficult to convince the regulator the goal had been achieved

Sizewell PPS safety arguments

- Mainly centred on the software
- Needed 10^{-4} *pfd*
- Safety case used evidence about quality of production, different kinds of assessment of built product (testing, extensive static analysis), etc
- This involved extensive expert judgment
- Regulators were not sufficiently confident in the 10^{-4} claim, but were prepared to accept 10^{-3}
- Eventually licensed for operation when the *secondary* system was judged to be an order of magnitude better than had been thought

This process prompted some questions

- How confident was regulator in original 10^{-4}?
- How confident was he in eventually-accepted 10^{-3}?
- How confident did he *need* to be?
- If his confidence in 10^{-3} is sufficiently high to be 'acceptable', how is this number used?
  - What happens to the residual uncertainty? (if he's 90% confident, what about the other 10%?)
- In fact there seemed to be an informal reasoning along the following lines: "we have some confidence - but not enough - in 10^{-4}, so let's only claim 10^{-3} *and treat this as if it were true*"
  - See our paper at DSN (Edinburgh, June 2007), for a way that such reasoning could be formalised

Don’t get me wrong…

…the regulators here were very good: honest and extremely competent

What do standards say?

- How confident in 10^{-9} have regulators been, when they have certified flight critical avionics?
  - What confidence does adherence to Do178B give us?
  - Nothing in the standard tells us (in fact it tells us nothing about the claim, let alone the confidence…)
- What is relationship between *claim* and *confidence* in, e.g., the SILs of IEC 61508?
  - You tell me…!
- Some standards *informally* acknowledge the problem
  - E.g. UK Def Stan 00-56 suggests use of a 'diverse two-legged argument' to *increase* confidence in a dependability claim
  - But it contains no guidance on issues concerning 'how much'

A simplistic illustration

Consider the case of operational testing of software. It is easy to show that if you have seen 4602 failure-free demands, you can claim that the *pfd* is smaller than 10^{-3} with 99% confidence.

- With the same evidence you could also claim $0.5*10^{-3}$ with 90% confidence, $0.15*10^{-3}$ with 50% confidence, and so on
- In fact there are an infinite number of (p, α) pairs for each set of evidence
- For any claim, p, you can always support it at *some* level of confidence
  - But would you be happy to fly in a plane when the regulator has said he is 0.1% confident that the flight control software has achieved the required 10^{-9}?

There are two sources of uncertainty…

- There is uncertainty about when a software-based system will fail
  - In the jargon: 'aleatory uncertainty'
  - It is now widely accepted that this uncertainty should be expressed probabilistically as a *dependability claim* (e.g. failure rate, *pfd*, etc)
- There is uncertainty about the reasoning used to support a dependability claim
  - In the jargon: 'epistemic uncertainty'
  - In particular, the role of *expert judgment*
  - The appropriate calculus here is Bayesian (subjective) probability
- This second type is largely ignored, or treated very informally
  - Particularly in our community (computer science, software engineering..)
  - Although there is a *nuclear* literature on the problem

Confidence-based dependability cases

If claims for dependability can *never* be made with certainty, we need a formalism that handles the uncertainty

- Informally, a dependability case is some reasoning, based on assumptions and evidence, that supports a dependability claim at a particular level of confidence
  - Sometimes convenient to deal with 'Doubt' = 1 - 'Confidence'
- For a particular claim (e.g. the probability of failure on demand of this system is better than 10^{-3}), your confidence in the truth of the claim depends on:
  - strength/weakness of evidence (e.g. the extensiveness of the testing)
  - confidence/doubt in truth of assumptions
  - correctness of reasoning
- Conjecture: assumption doubt is a harder problem to handle than evidence weakness

An example

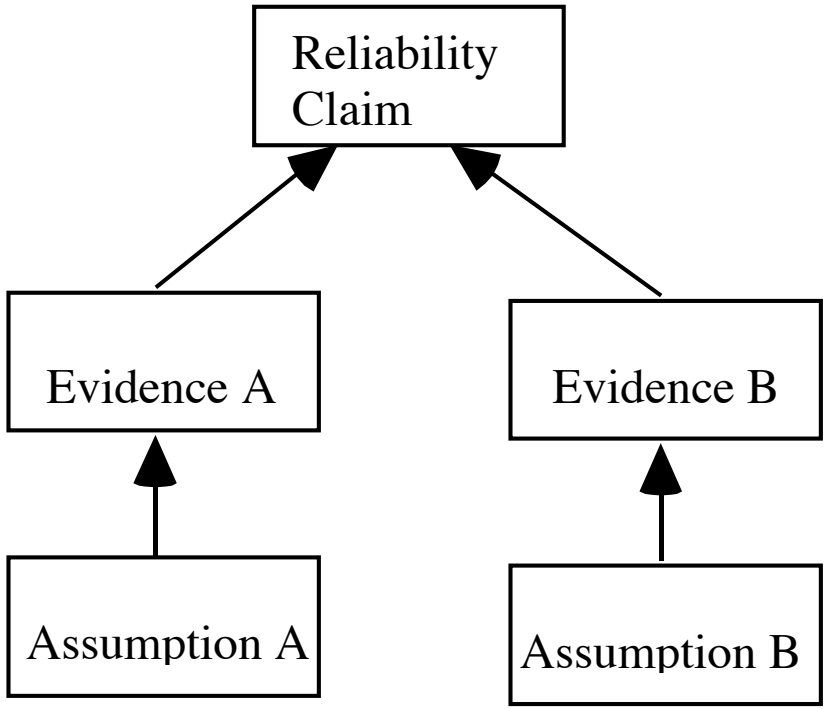
The following example from our recent work illustrates how confidence can treated formally as part of a dependability case

- And how there can be unexpected pitfalls
- **For details, see our paper in May 2007 issue of *IEEE Trans Software Engineering***

Dependability case 'fault tolerance'

Can we borrow ideas from *system* fault tolerance? 'Argument diversity' as analogy of 'system diversity'?

- Multi-legged arguments to increase confidence in reliability claim(s)
  - leg *B* could overcome evidence weakness and/or assumption doubt in leg *A*
  - legs need to be *diverse*
  - advocated in some existing standards (but only informal justification)



Motivation: analogy from *systems*

- the use of diverse redundancy to mask failure is ubiquitous
  - 'two heads are better than one', 'belt *and* braces', 'don't put all your eggs in ne basket'
  - e.g. scientific peer review; e.g. multiple medical diagnoses
- commonly used for *systems*
  - e.g. design-diverse critical software in Airbus aircraft
- often used in software development *processes*
  - e.g. diverse procedures to find software faults
- reasonably good understanding of these applications of diversity
  - e.g. formal probability models
- do these ideas work for *dependability cases*?

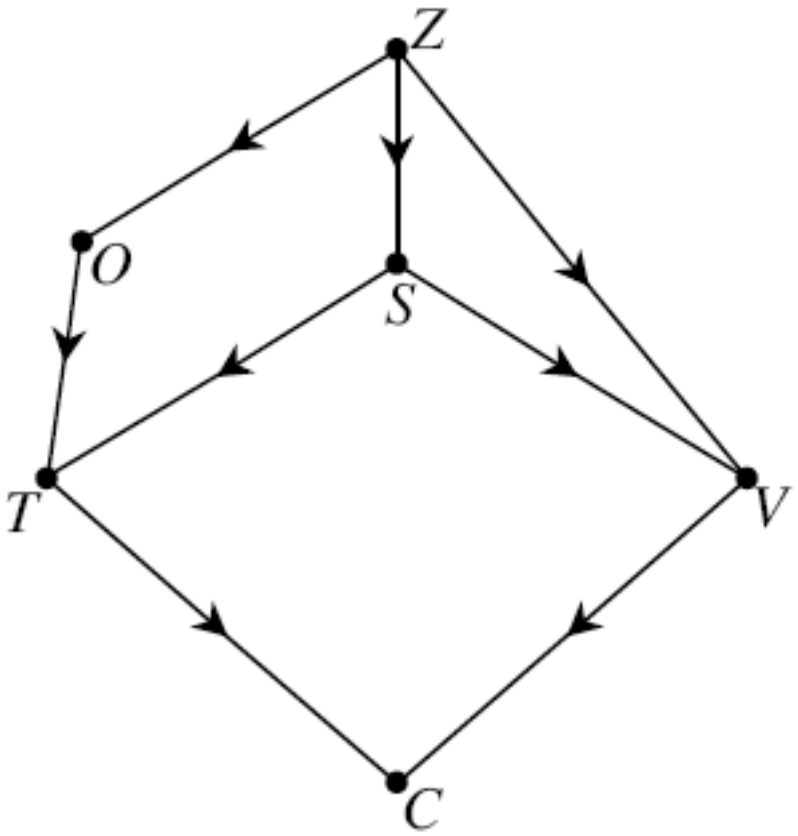
Do multi-legged arguments increase confidence? If so, how much?

We have examined a simple *idealised* example in some detail.

- motivated by (relatively) simple software for a protection system
- two argument legs
  - testing
  - verification
- dependability claim is '*pfd* is smaller than 10^{-3}'

Our approach uses BBN models of the arguments, which are manipulated *analytically* via parameters that determine their node probability tables (compared with more usual purely numeric approach to BBNs)

2-legged BBN topology



S: system's true unknown pfd, 0≤S≤1

Z: system specification, {correct, incorrect}

O: testing oracle, {correct, not correct}

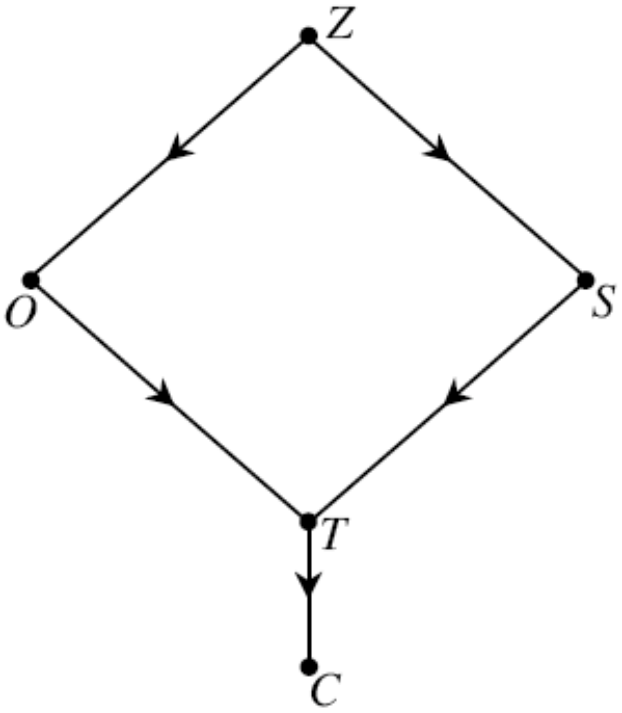
V: verification outcome, {verified, not verified}

T: test result, {no failures, failures}

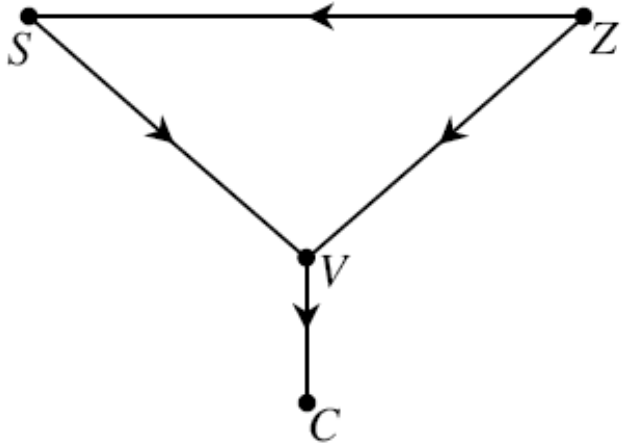
C: final claim, {accepted, not accepted}

(V,T) represents what we have called 'evidence'. We shall only consider the 'perfect outcome' situation here: T=no failures, V=verified

Single leg topologies



Testing leg



Verification leg

Computations with this BBN

We are interested in how good 2-legged arguments are - for example, in how much the 2-legged argument improves on the single arguments

- E.g. could evaluate confidence in claim, $P(S \leq 10^{-3} \mid VT)$, and compare with $P(S \leq 10^{-3} \mid T)$ and $P(S \leq 10^{-3} \mid V)$
  - how much better is 2-legged argument than each of the single-legged arguments?
- E.g. we could evaluate $P(CS \mid \text{evidence}) = P(CS \mid VT)$
  - in particular $P(C = \textit{accepted}, S > 10^{-3} \mid \text{evidence})$, concerning unsafe failure of an argument

These involve elicitation of complex prior beliefs (to fill in the *node probability tables* of the BBNs)

- This involves extensive expert judgment - as is usual with software-based systems

The model is complex…

…in spite of its idealisation. So we make simplifying assumptions (our aim is to be conservative). We can then manipulate the resulting mathematics, e.g. doubt about *pfd* claim is

$$\frac{\xi(1-p_{0|c})\left[\pi_{cc}\mu' I_{1-s}(b'+n,a')+\pi_{ci}I_{1-s}(b',a')\right]+(1-p_{0|i})\left[\pi_{ic}\mu I_{1-s}(b+n,a)+\pi_{ii}I_{1-s}(b,a)\right]}{(1-\alpha)p_{0|c}\pi_{c*}+p_{0|i}\pi_{i*}+\xi(1-p_{0|c})\left[\pi_{cc}\mu'+\pi_{ci}\right]+(1-p_{0|i})\left[\pi_{ic}\mu+\pi_{ii}\right]}$$

- (I'm not going to talk about details of the maths!)
- parameters here capture different aspects of prior belief
- advantage over purely numerical approach to BBNs is that we know what the parameters *mean*
- and you can do 'what if' calculations using MathCad, Mathematica, Maple etc

Surprise 1

Evidence that is *supportive* (i.e. 'clearly' good news) can *decrease* confidence, even in a *single argument leg*!

- Example: Testing leg. We have a set of parameters (i.e. beliefs) for which seeing very many failure-free test cases (>17,000) decreases confidence from *a priori* value of 0.99583 to 0.66803
- Seems counter-intuitive, but is it? Key role is played by 'assumption doubt', and how this changes as we see evidence (here lots of failure-free operation)
- This centres on the matrix, ***P*(*Z*, *O*)**:

		O: correct	O: incorrect	
Z	correct	π_{cc}	π_{ci}	π_{c*}
	incorrect	π_{ic}	π_{ii}	π_{i*}
		π_{*c}	π_{*i}	1

Surprise 1 (contd.)

The assumption doubt changes as follows:

$$P(ZO) = \begin{bmatrix} 0.99419 & 1.63910\times10^{-3} \\ 7.81537\times10^{-5} & 4.09042\times10^{-3} \end{bmatrix}$$

$$P(ZO\,|\,T) = \begin{bmatrix} 0.53406 & 0.13329 \\ 1.27237\times10^{-5} & 0.33263 \end{bmatrix}$$

- Informally: seeing no failures could be evidence for small *pfd*, or for defective oracle
  - reasonable that *Z, O, S* prior beliefs are positively associated
  - so increased doubt about oracle, as here, can imply increased doubt about *S*
- We call arguments like this, that reduce confidence, '*non-supportive*'
  - what is surprising is that they can be based on *supportive* evidence

Surprise 2 (the big one!)

- What happens with 2-legged arguments?
- If you add a supportive argument to an existing argument does your confidence increase?
- *Sometimes not!*
- This arises, again, from a subtle 'backstairs' inferential chain of reasoning
  - See our paper for an after-the-fact intuitive explanation
  - But note that this was not obvious before we did the detailed formal analysis - it surprised us!!
- Notice how all this contrasts with *systems*, where a 1-out-of-2 system is *always* better than each single channel

Discussion

What does all this mean?

- could we expect these counter-intuitive results to occur in practice?
  - not sure, but difficult to justify ruling this out
  - do the results arise from our model simplifications?
    - + we think not, but cannot be sure
- on the other hand, we have seen plausible beliefs for our model which do *not* result in these counter-intuitive results
  - e.g. get respectable increase in confidence from adding a second argument leg
  - argument diversity (sometimes) works

Discussion (2)

- At least, there is a possibility for subtle interactions between beliefs/assumptions/confidence when dealing with disparate evidence in dependability cases
  - naïve purely-numeric BBN results need to be treated with suspicion
  - human judgment, unaided by a formalism, even more so?
- We have demonstrated the feasibility of a formal analysis of these kinds of dependability cases in terms of claim-confidence
  - can show consequences of *a priori* beliefs to experts
  - give feed-back
- However, it gets very hard to do this for realistic arguments
  - we have some ideas about how to relax some of our simplifications
- There are some difficult issues concerning elicitation of belief from experts in practice

So where does all this leave us?

In this talk I wanted to make two main points:

- There is a *need* for quantitative dependability cases, based on a formal calculus of confidence
- This can be provided via formal (Bayesian) probabilistic modelling

On the *need* for a theory of confidence

- Some years ago, a regulator told me: "Yes, I do believe the A320's flight control system is a 10^{-9} system"
- I've seen a railway signalling system where the apparent requirement is a failure rate no bigger than 10^{-12} per hour!
- I believe that confidence in such claims, based on rigorous arguments, would be very low
  - Responsibility lies with the builders of such systems to demonstrate high confidence in such a way that this can be agreed by third parties
  - And if this can't be done for a safety-critical system, should it be allowed to be deployed?

On the *need*….(2)

But it's not all gloom

- Even for critical systems, ultra-high reliability figures are rare
  - E.g. the Sizewell PPS figure is quite modest: it should be possible to obtain high confidence in such a figure
- These comments do not only apply to *critical* systems: for other systems, a confidence-based approach would be valuable
  - E.g. the need for banks to assess IT risks under the Basel II accords

The Bayesian approach needs more work

For example, we need a much more holistic approach

- Beyond 'software and computers'
  - it's very rare for systems to be purely 'technical' - there are almost *always* humans and organisations involved, and the *whole system* needs to be addressed
  - interactions here can be complex and counter-intuitive
  - require collaboration with psychologists, sociologists, etc
- Beyond 'reliability and safety', to incorporate *security*
  - very little work has been done on problem of (probabilistic) *security assessment*
  - but some of the reliability techniques probably apply
  - need to be able to understand trade-offs

But (and finally) beware simple panaceas

- There are deep subtleties in the relationships between the constituents of dependability arguments (assumptions, evidence, reasoning, claims, confidence)
  - These seem to be inherent - you can't wish them away
  - You ignore them at your peril
- Unaided expert judgment could get things badly wrong
  - Even BBNs, when these are simply numeric, can be very misleading and lead to misplaced trust

Thank you for listening!

(I'm assuming at the time of writing that you will have been!)

Questions?

Brickbats?