PROBABILITY MODELS FOR DESIGN DIVERSITY - AND FOR OTHER MANIFESTATIONS OF DIVERSITY

B Littlewood

Rapporteur: Dr L B Arief

Probability models for design diversity - and for other manifestations of diversity

Bev Littlewood
City University
London EC1V 0HB
b.littlewood@csr.city.ac.uk
Phone: +44 020 7477 8420

diversity *n.* & *a.* ME. [(O)Fr. *diversité* f. L. diversitas, *f.* diversus DIVERS: see -ITY] A *n.* 1 The condition or quality of being diverse, different, or varied; variety, unlikeness. 2 An instance of this .....

- The New Shorter Oxford English Dictionary

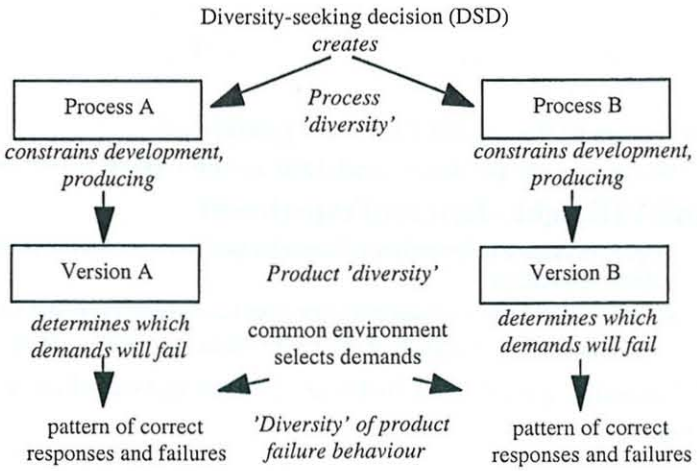
Design diversity

- Separate ('independent') creation of 2 or more versions of a program (or, more generally, a wider design involving hardware and software)
  - could have common specification; could be spec.-diverse
  - teams work 'independently' of one another
  - may be forced to use different methods ('forced diversity')
- At run-time, there is adjudication among the different version outputs to produce a single 'best guess'
  - e.g. in N-version programming, could be a 2-out-of-3 voter
  - e.g. acceptance test in Recovery Block architecture
  - e.g. 1-out-of-2 actuation in some nuclear safety systems
- We hope diversity of design implies diversity of version failures

Intuitive technical rationale

- 'Two heads are better than one': this kind of reasoning is ubiquitous in many areas of human activity
  - in computing, Babbage was probably the first to advocate 'using two computers' (*but he meant people!*)
- Hardware *redundancy* was an early analogy
  - 'make arbitrarily reliable systems from arbitrarily unreliable components'
  - the analogy can be very misleading - claims for independence of failures are not believable for software
- *Functional diversity* in hardware systems is a better analogy
  - but we shall see that claims for independence here are difficult to sustain

How do we get design diversity?



Design diversity - what's the evidence?

- **Industrial use examples:**
  - railway signalling: U.S., Sweden, Italy, Austria
  - aerospace: mostly Airbus, but also Boeing, NASA (Space Shuttle)
  - nuclear: Temelin protection system
- **Research case studies:**
  - several in the nuclear industry
  - Newcastle command-and-control system
  - aerospace
- **Multiple developments and statistical experiments:**
  - several commissioned by NASA

Historical fortunes

- Industrial applications, with general satisfaction
  - but systems probably too good to *show* reliability gain
  - and this may be substantial
- *But* some claims of failure independence!
  - 'if you need 10^{-6} pfd, build parallel system with versions having 10^{-3}'
- NASA (Knight - Leveson) experiment
  - disproves failure independence, indicates positive correlation (though gain in reliability)
  - some strong reactions from industry, regulators (and academics!)
  - resulted in unfair dismissal of the approach in several industries
- What sort of numbers did the experiments and case studies show?

Efficacy: case study example

- Naval command and control demonstrator - Anderson *et al* in Newcastle
  - recovery blocks (simple application)
  - 74% potential failures averted by fault tolerance
  - costs:
    - 60% increase in development costs
    - 33% extra code memory
    - 35% extra data memory
    - 40% additional run-time

Example: Knight - Leveson experiment

- problem: (small part of) missile defence system
- 27 versions submitted to 1,000,000 tests
- 1,255 'common mode' failures
- At most 8 versions failed together
- hypothesis 'all versions fail independently' rejected with extremely high confidence
- 2-out-of-3 voting 19 times better (more reliable) than single version on average
  - but huge variation between individual versions and between individual triplets (e.g. best 'singles' better than worst 'triples')

Need for better understanding

- Seems we must assume failure dependencies
- Goal for reliability *achievement*: we would like, e.g., P(*A* and *B* fail together) to be as small as possible
- Goal for reliability *assessment*: we would like to know what this failure probability actually *is*
- Requires more formal, quantitative, understanding of roles of 'independence', 'diversity', etc
  - e.g. trade-offs between 'version reliability' and 'version dependence'
  - e.g. *natural* versus *forced* diversity (e.g. via process, via people)
- Eckhardt and Lee (EL), later generalised by Littlewood and Miller (LM), were first formal probability models for diversity

The EL model (1)

- Introduces the notion of *variation of difficulty* over the input space
- There is a *set of all programs* that might be written to a particular specification, and a *probability of selection* (i.e. creation) for each
- There is a *set of all possible inputs*, and a *probability of selection* for each
- Execution of, say, a 1-out-of-2 diverse system involves random selection (i.e. software development) of 2 programs from the set of all programs, and then the random selection of an input.

The EL model (2)

- Key measure is $\theta(x)$, the probability of failure of a randomly chosen program on a *particular* input, x
- EL say that $\theta(x)$ takes different values for different x - i.e. some inputs are 'more difficult' than others (more programs would fail on them)
- For a randomly chosen input, this is a random variable, Θ; the distribution of this is the heart of the model
- The mean of the distribution, $E(\Theta)$, is the *probability that a randomly chosen program fails on a randomly chosen input* - it is the (un)reliability of a randomly selected program

The EL model (3)

- Now select (i.e. develop) 2 program versions, A, B, independently
- It turns out that
  P(A, B both fail on randomly chosen input) = $E(\Theta^2)$
  = $[E(\Theta)]^2 + Var(\Theta)$
- This is *bad news!* First term is 'independence' result
- P(A, B both fail on randomly chosen input)
  ≥P(A fails).P(B fails)
- Assumption of independence is *always* too optimistic
  - error depends on *how much* variation of difficulty there is
  - the problem arises from the fact that *conditional independence* does not imply *unconditional independence*

Intuitive explanation

- You observe version A fail on a particular input
- You conclude "this was probably a 'difficult' input"
- You therefore think there is an increased chance of B failing (the input will also be difficult for B)
  - *even though there is (conditional) independence between A and B failures*
- 'Independent programs' (programs developed independently) do not fail independently
- the *mean* and *variance* of the difficulty function are key to understanding

The LM model (1)

- 'Difficulty' depends on application demand but also on software process, general design, staff..: *'methodology'*
- EL describes multi-version, single methodology software
- LM models multi-version, *multiple methodology* programming: think of it as *'forced diversity'*
- Consider one version selected randomly from methodology A, one from methodology B
- Separate difficulty functions $\theta_A(x)$, $\theta_B(x)$
- Probabilities of demands are the same for both versions

The LM model (2)

- Now the probability that both (randomly chosen) programs fail on randomly chosen demand is

  P(A and B fail)

  $= E(\Theta_A).E(\Theta_B) + Cov(\Theta_A, \Theta_B)$

  - first term on right is naive "independence" answer
- Covariance: intuitively plausible measure of similarity of difficulty functions
- It follows that

  P(B fails | A failed) > P(B fails)

  if and only if $Cov\ (\Theta_A, \Theta_B) > 0$

  - it is possible to do better than independence of behaviour (and even this is unattainable in EL set-up)

What do the models tell us?

- They provide a formalism for understanding diversity
  - involves first and second moments of difficulty functions
    - means = (un)reliabilities
    - variances, covariances = diversity, dependence
  - these can be difficult or impossible to estimate in practice, though
- We need to worry not only about people making the *same* mistakes
  - key idea is varying propensity for human error
  - programs will tend to fail in the same circumstances, but in possibly different ways
- Simple calculations of system reliability assuming *independence* are (essentially) never valid

What do the models tell us? (2)

- We can prove that, subject to certain conditions, diversity is always 'a good thing'
  - AB better than AA or BB, if you're indifferent between the latter
- We can prove that, all things being equal, *more* diversity is better than *less*
  - we have formal measure of 'amount' of diversity - a 'distance' between methodologies
- If we have many sources of diversity, we can (in principle) deploy it in most effective way
- BUT the difficulty in all of this is estimating the key model parameters
  - means that many of the insights are conceptual or qualitative rather than quantitative

Applications outside *design* diversity

- The key modelling idea - 'difficulty variation' - is applicable to other kinds of diversity
  - e.g. diversity of fault-finding methods for software debugging
    - interestingly, there is promise here that the key parameters can be estimated from real-life data
  - e.g. human/computer diversity
    - promising work in DIRC by Andrey Povyakalo, Lorenzo Strigini
  - e.g. diverse 'multi-legged' arguments for safety cases
- I'll talk about the first of these in detail on Thursday, with numerical examples

Diverse arguments

- Actually used in some safety cases
  - e.g. Sizewell B nuclear plant Primary Protection System (PPS) software
- 2 (or more) arguments used to support a claim
  - e.g. for PPS software, one argument based on 'quality of build', one based on 'assessment of built product'
  - here claim was 10^{-3} pfd
- Each argument sufficient in principle to support claim
  - but there are fears of 'defects' in each argument leg
- Reasoning of this type has been rather informal
  - 'we feel more confident with two legs than we would with either one alone'

Can we formalise this?

- In particular, do the probabilistic diversity models apply to this?
  - e.g. are there traps for the unwary here, as there were in design diversity?
- One approach: treat 'confidence' in an argument like 'reliability' was treated for a designed system
  - e.g. if each leg allows us to claim 'it's a 10^{-3} pfd system' with 99% confidence, do the two together allow a claim at 99.99% confidence?
- Seems likely that such independence assumptions are just as suspect here as they are elsewhere
  - so what *can* be claimed?

Possibilities....

- Can we force argument diversity?
  - choose evidence, and structure arguments, to maximise the difference
- How do we handle the trade-off between the individual argument 'strengths' (confidence they induce), and the diversity between them?
- Can we build a *quantitative* theory of diverse arguments?
- Safety cases for software-based systems currently have a large element of expert judgement - how does this fit in?
  - need psychological input: looking at this in DIRC

Finally, a different sort of diversity

- The following argument was suggested by the safety arguments for the Sizewell B protection system
- They needed 10^{-7} pfd overall
- 1-out-of-2 system: complex software-based PPS and simpler hard-wired SPS
- Safety case reasoning as follows:
  - claim 10^{-x} for PPS, 10^{-y} for SPS
  - assuming independence implies $10^{-(x+y)}$ for overall protection function
  - BUT they did not trust this independence argument
  - SO they decided to downgrade x and y
  - ended up claiming 10^{-7} from 10^{-3} and 10^{-4}
  - strange trade-off between *version reliability* and *dependence* between versions

Primary + simple secondary

- Sizewell is an example of a common 'asymmetric' architecture: extensive functionality in a primary system, plus much simpler secondary
  - e.g. in aircraft: sophisticated primary flight control system for 'normal' use, plus simple 'get you home' back-up
- Seems a sensible way to *achieve* reliability - does it help to justify a reliability *claim*?
- Can we combine a claim for *reliability* in the primary, with claim for *perfection* in the secondary?
  - i.e. combine *measurement* (of reliability) with *proof* (of perfection)
- *Measurement* and *proof* seem very diverse (to me!)

Primary + simple secondary (2)

- **Assume conservatively that if SPS not perfect, it always fails when PPS does:**

$$
\begin{aligned}
&P(\text{safety system fails}) \\
&= P(\text{safety system fails} \mid \text{SPS perfect})P(\text{SPS perfect}) \\
&+ P(\text{safety system fails} \mid \text{SPS not perfect})P(\text{SPS not perfect}) \\
&= P(\text{PPS fails} \mid \text{SPS not perfect})P(\text{SPS not perfect})
\end{aligned}
$$

- **If we are prepared to assume *independence* between failure of PPS and imperfection of SPS:**

$$P(\text{safety system fails}) = P(\text{PPS fails})P(\text{SPS not perfect})$$

Primary + simple secondary (3)

- **Independence here is more plausible than it is between failures of complex versions**
  - **e.g. for software, possibility that verification of SPS is incorrect seems independent of whether PPS fails**
- **Even if independence cannot be claimed, it may be possible to make a conservative claim for**
  $P(\text{PPS fails} \mid \text{SPS not perfect})$
- **Need to have estimate of probability of correctness**
  - **cannot be measured**
  - **expert judgement? conservative judgement?**
- **Would this have worked for Sizewell?**
  - **need 10^{-3} for PPS pfd, 10^{-4} probability of SPS imperfection**

Conclusions

- Naïve independence claims based on design diversity never believable
- But strong evidence of benefits - issue is 'how much?'
- Models give better understanding of nature of dependence/diversity
  - particularly trade-offs, e.g. between version reliability and diversity
  - some results intuitively 'obvious', others not so
  - beginnings of a quantitative calculus
- Potential for wider applications of diversity
  - diverse SE processes, human/computer diversity, etc
- Can diversity be employed to aid both *achievement* and *assessment* of dependability?

DISCUSSION

Rapporteur: Dr L.B. Arief

Lecture One

Regarding general evaluation of design diversity in industrial application, Professor Jones wondered whether it is due to the lack of instrumentation. Professor Littlewood replied that although there is no legal requirement for it, instrumentation does exist, for example in the Airbus 300 or 310 series.

Professor Martin commented that there might be some fine gradation on the students involved in the Knight-Leveson experiment, which concluded that the best single version is better than the worst triple version. Professor Littlewood answered that there is no distinction among the students.

On the subject of probability of failure depending on input, Professor Martin gave an example on stack overflow. Professor Littlewood pointed out that he is not a computer scientist, so he is not familiar with stack overflow in programs. Professor Schneider then indicated that bigger input might cause stack overflow to happen.

Professor Schneider argued that making two programs worse (by making them fail more often) might reduce the variance, but this way seems counter-intuitive that it should result in better expectation. Professor Littlewood replied that they are different things, one is the average value of the difficulty function and the other is how much the difficulty function varies. So, it is possible to have quite different variances for the same average value.

Concerning the Sizewell protection system, Dr Rushby wanted to know what it means to be primary or secondary protection system, as it seems like both are able to shut down the system. Professor Littlewood agrees that both are able to do that, but the primary is the one that should do it because it shuts down the system gently. So there are economic reasons for favouring it.

Professor Suri questioned how much of diversity will be useful. Professor Littlewood tried to understand the question as how much of trade off between version reliabilities and diversity (between versions) would be beneficial. It is probably often the case that when diversity is increased, the version reliabilities might be reduced. So there are a lot of problems about the interplay between version reliability and diversity, and we need to know all about those in order to talk about system reliability.