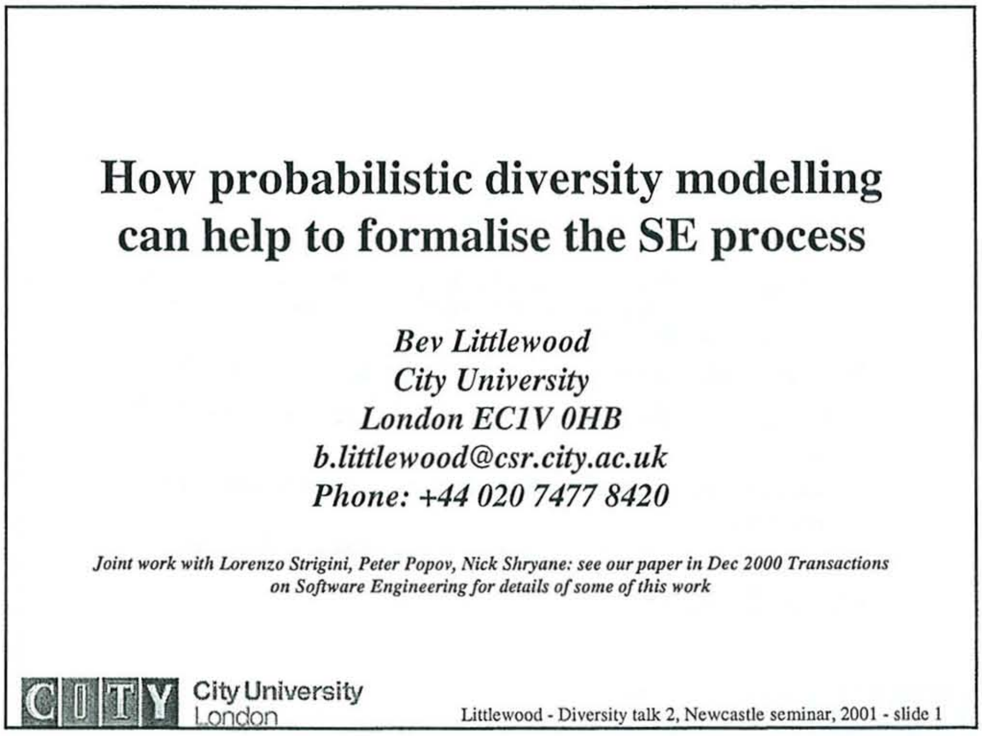
HOW PROBABILISTIC DIVERSITY MODELLING CAN HELP TO FORMALISE THE SE PROCESS

B Littlewood

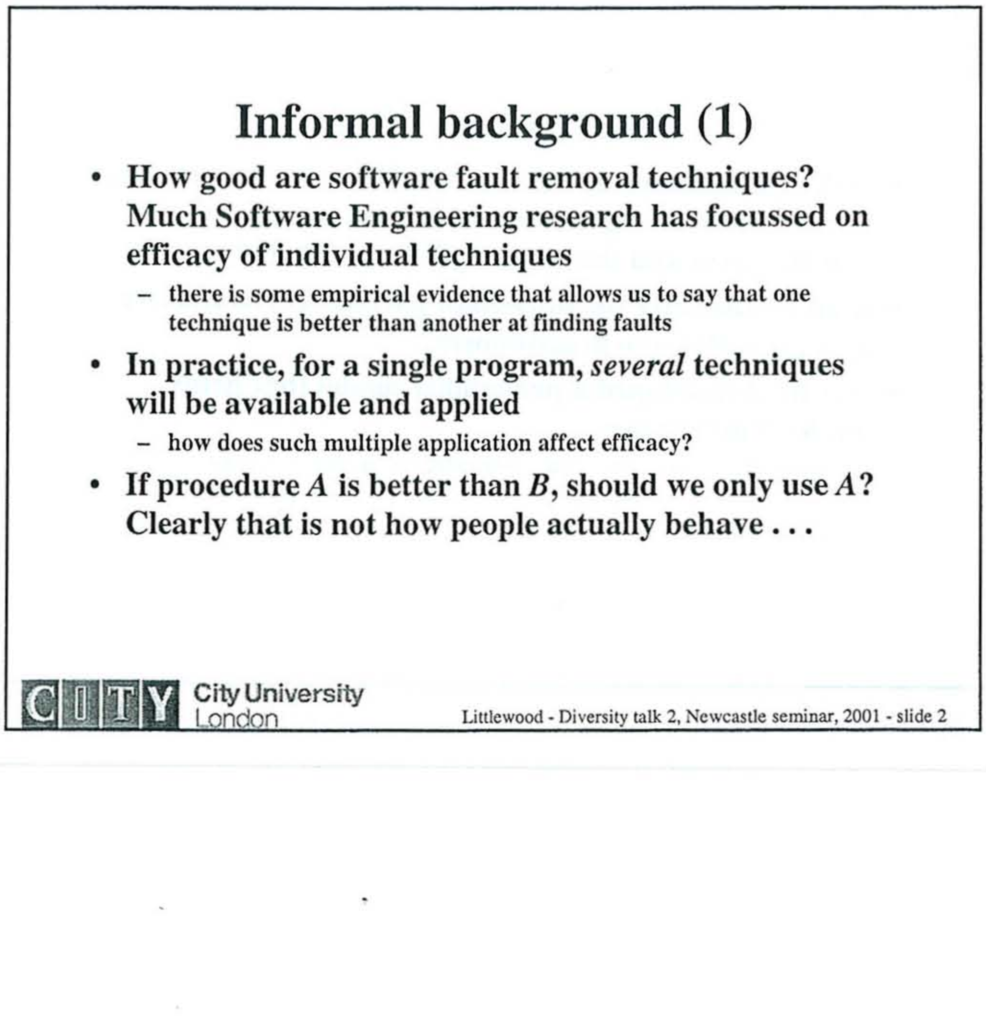
Rapporteur: Dr L B Arief



How probabilistic diversity modelling can help to formalise the SE process

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

Joint work with Lorenzo Strigini, Peter Popov, Nick Shryane: see our paper in Dec 2000 Transactions on Software Engineering for details of some of this work



Informal background (1)

- **How good are software fault removal techniques? Much Software Engineering research has focussed on efficacy of individual techniques**
  - there is some empirical evidence that allows us to say that one technique is better than another at finding faults
- **In practice, for a single program, *several* techniques will be available and applied**
  - how does such multiple application affect efficacy?
- **If procedure *A* is better than *B*, should we only use *A*? Clearly that is not how people actually behave . . .**

Informal background (2)

- **Informally: 'don't put all your eggs in one basket'**
  - **different procedures will target different types of faults with different efficacy**
    - **e.g. in practice we may decide to stop applying procedure *A* (e.g. static analysis) and apply procedure *B* (e.g. testing), even though we know *A* is better than *B* overall**
- **It is not only the *effectiveness* of the procedures that matters, but also their *diversity* (i.e. how 'different' they are)**
  - **we need to understand this interplay between *effectiveness* and *diversity***
- **This work inspired by earlier probability modelling for *design diversity***

City University London

The key questions

- **What is the best way to use the different fault removal procedures, taking account of their individual effectiveness and the diversity between them?**
- **Can we measure the important parameters so that we can give advice to practitioners?**
- **We have developed a probability model that helps understanding here**
  - **formally, it is exactly similar to models for design diversity**

City University London

Our model

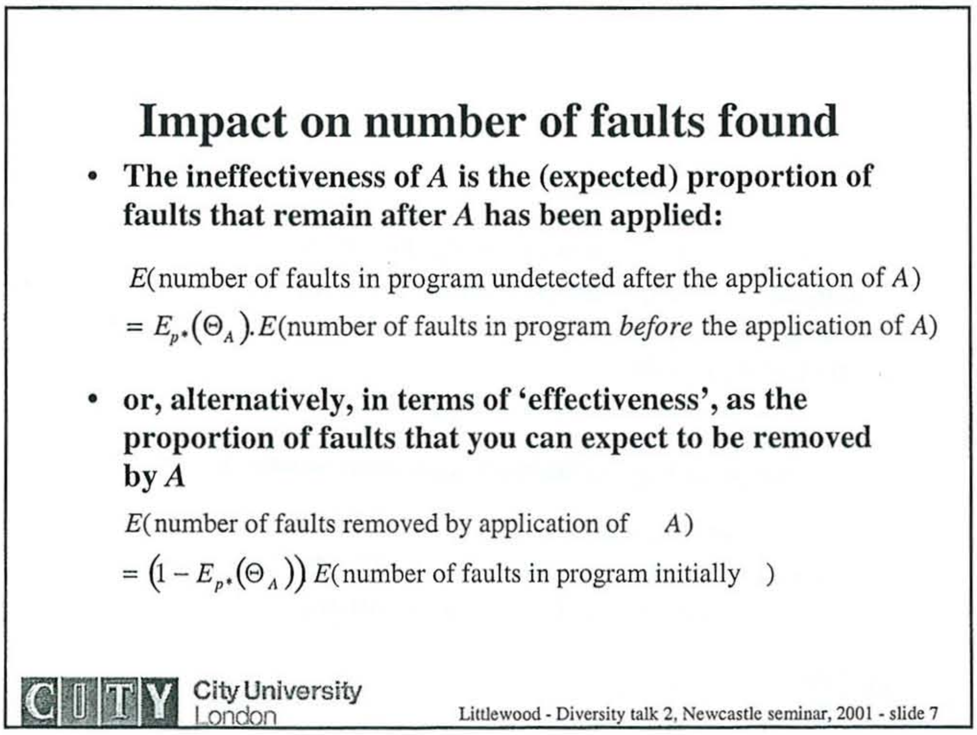
- When a particular fault-finding procedure is applied, there is *uncertainty* about the outcome
  - each fault may or may not be successfully detected
- Key idea is a *difficulty function* for each procedure, *A*. For a fault *i* this is the probability that *A* will *not detect* the fault, $\theta_A(i)$
  - we can think of this as the difficulty you would have in detecting fault *i* using procedure *A*
  - the important point is that this will take different values for different faults
    - some faults are harder to detect (by *A*) than others
- We have, in addition, probability distributions:
  $p_i^* = P(\text{randomly selected fault is of type } i)$

City University London

'Ineffectiveness'

- We define '*ineffectiveness*' of *A* as
  $P(A \text{ fails to detect a randomly chosen fault })$
  $$= \sum_i p_i^* . \theta_A(i) = E_{p^*}(\Theta_A)$$
- This can be thought of as the 'unreliability' of the procedure at finding faults
  - it is the average 'difficulty' over all faults
- It represents the ineffectiveness of procedure *A* in detecting faults in the following intuitively appealing and precise way:

City University London



Impact on number of faults found

- **The ineffectiveness of *A* is the (expected) proportion of faults that remain after *A* has been applied:**

E(number of faults in program undetected after the application of A)

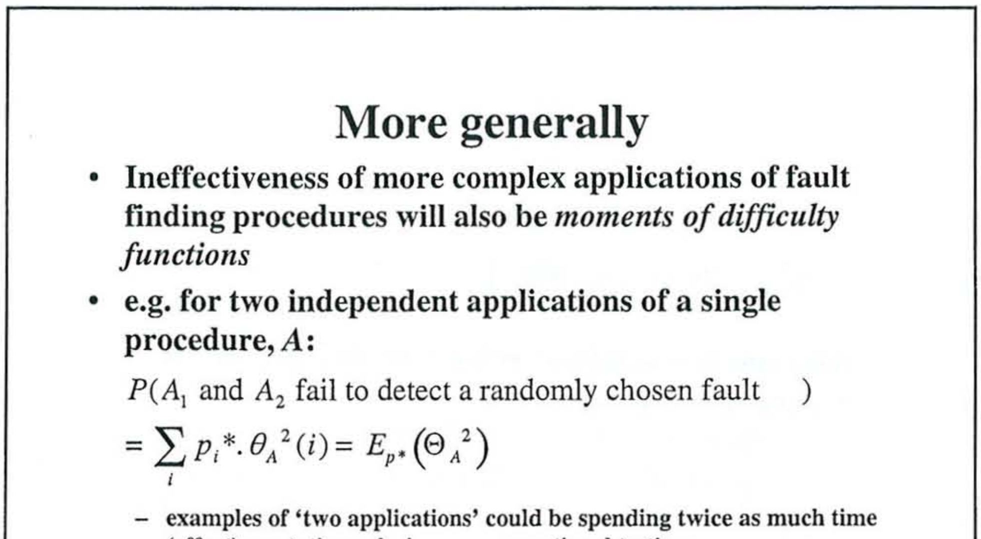
$= E_{p^*}(\Theta_A).E$(number of faults in program *before* the application of A)

- **or, alternatively, in terms of 'effectiveness', as the proportion of faults that you can expect to be removed by *A***

E(number of faults removed by application of A)

$= (1 - E_{p^*}(\Theta_A))\, E$(number of faults in program initially)

City University London



More generally

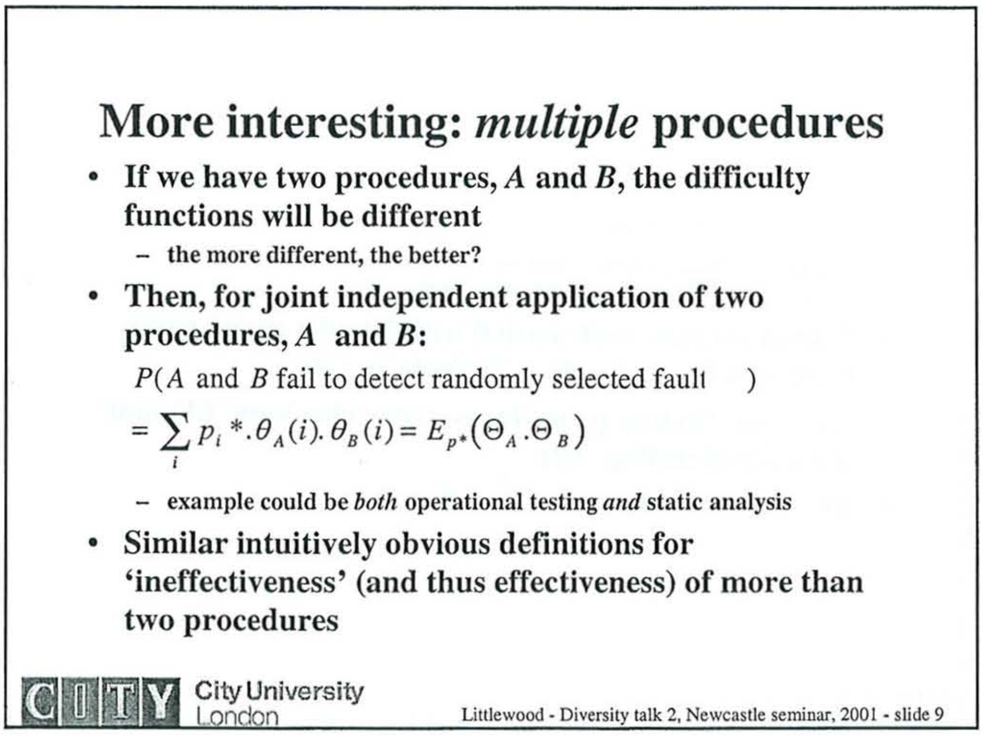
- **Ineffectiveness of more complex applications of fault finding procedures will also be *moments of difficulty functions***
- **e.g. for two independent applications of a single procedure, *A*:**

$P(A_1$ and A_2 fail to detect a randomly chosen fault)

$$= \sum_i p_i^*.\theta_A^{\,2}(i) = E_{p^*}(\Theta_A^{\,2})$$

  - **examples of 'two applications' could be spending twice as much time (effort) on static analysis, or on operational testing**

City University London



More interesting: *multiple* procedures

- **If we have two procedures, *A* and *B*, the difficulty functions will be different**
  - **the more different, the better?**
- **Then, for joint independent application of two procedures, *A* and *B*:**

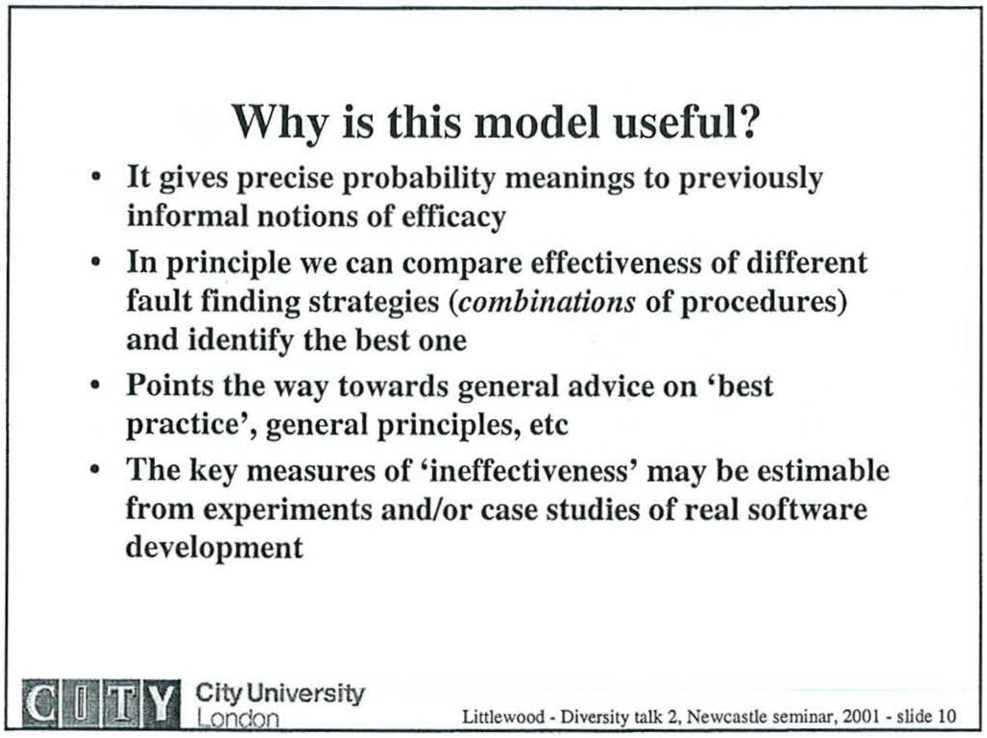
  $P(A$ and B fail to detect randomly selected fault $)$

  $= \sum_i p_i{*}.\theta_A(i).\theta_B(i) = E_{p*}(\Theta_A.\Theta_B)$

  - **example could be *both* operational testing *and* static analysis**
- **Similar intuitively obvious definitions for 'ineffectiveness' (and thus effectiveness) of more than two procedures**

City University London

Littlewood - Diversity talk 2, Newcastle seminar, 2001 - slide 9

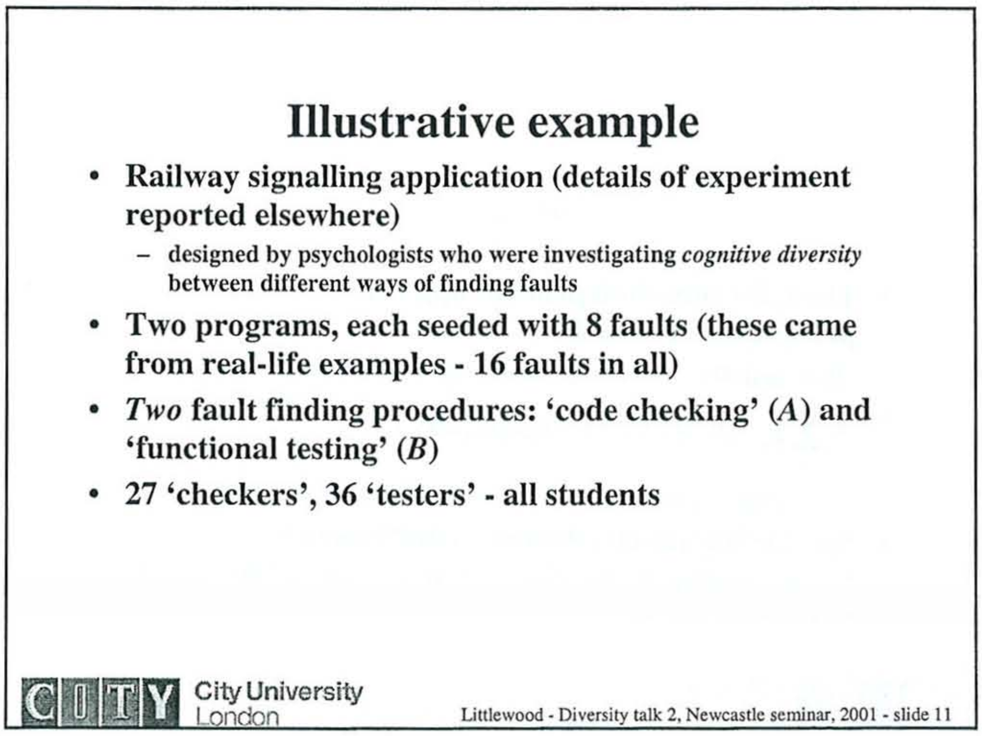


Why is this model useful?

- **It gives precise probability meanings to previously informal notions of efficacy**
- **In principle we can compare effectiveness of different fault finding strategies (*combinations* of procedures) and identify the best one**
- **Points the way towards general advice on 'best practice', general principles, etc**
- **The key measures of 'ineffectiveness' may be estimable from experiments and/or case studies of real software development**

City University London

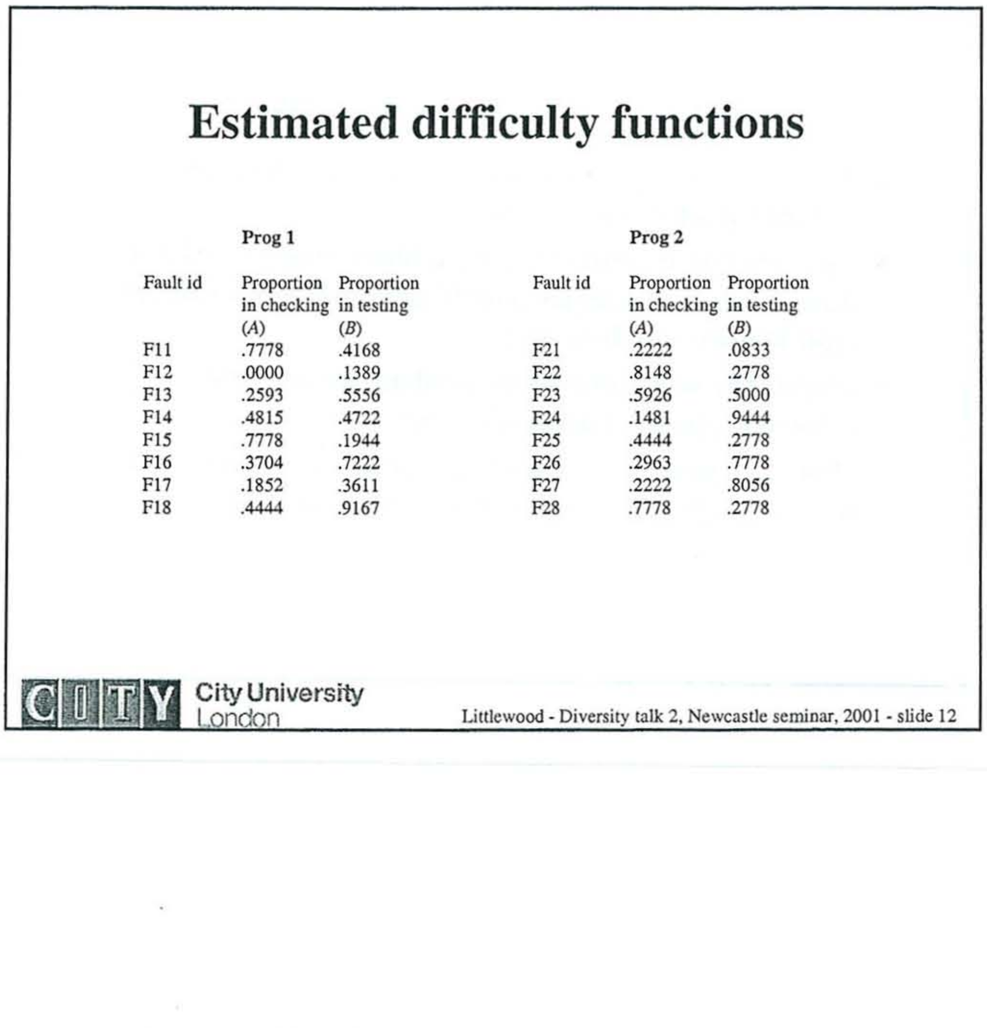
Littlewood - Diversity talk 2, Newcastle seminar, 2001 - slide 10



Illustrative example

- Railway signalling application (details of experiment reported elsewhere)
  - designed by psychologists who were investigating *cognitive diversity* between different ways of finding faults
- Two programs, each seeded with 8 faults (these came from real-life examples - 16 faults in all)
- *Two* fault finding procedures: 'code checking' (*A*) and 'functional testing' (*B*)
- 27 'checkers', 36 'testers' - all students

City University London



Estimated difficulty functions

	Prog 1			**Prog 2**	
Fault id	Proportion in checking (*A*)	Proportion in testing (*B*)	Fault id	Proportion in checking (*A*)	Proportion in testing (*B*)
F11	.7778	.4168	F21	.2222	.0833
F12	.0000	.1389	F22	.8148	.2778
F13	.2593	.5556	F23	.5926	.5000
F14	.4815	.4722	F24	.1481	.9444
F15	.7778	.1944	F25	.4444	.2778
F16	.3704	.7222	F26	.2963	.7778
F17	.1852	.3611	F27	.2222	.8056
F18	.4444	.9167	F28	.7778	.2778

City University London

Results for Program 2

$E_{p^*}(\Theta_A) = 0.440, \quad E_{p^*}(\Theta_B) = 0.493$

$E_{p^*}(\Theta_A^{\,2}) = 0.253, \quad E_{p^*}(\Theta_B^{\,2}) = 0.329$

$E_{p^*}(\Theta_A.\Theta_B) = 0.179, \quad E_{p^*}(\Theta_A)E_{p^*}(\Theta_B) = 0.217$

$Cov_{p^*}(\Theta_A, \Theta_B) = -0.0381$

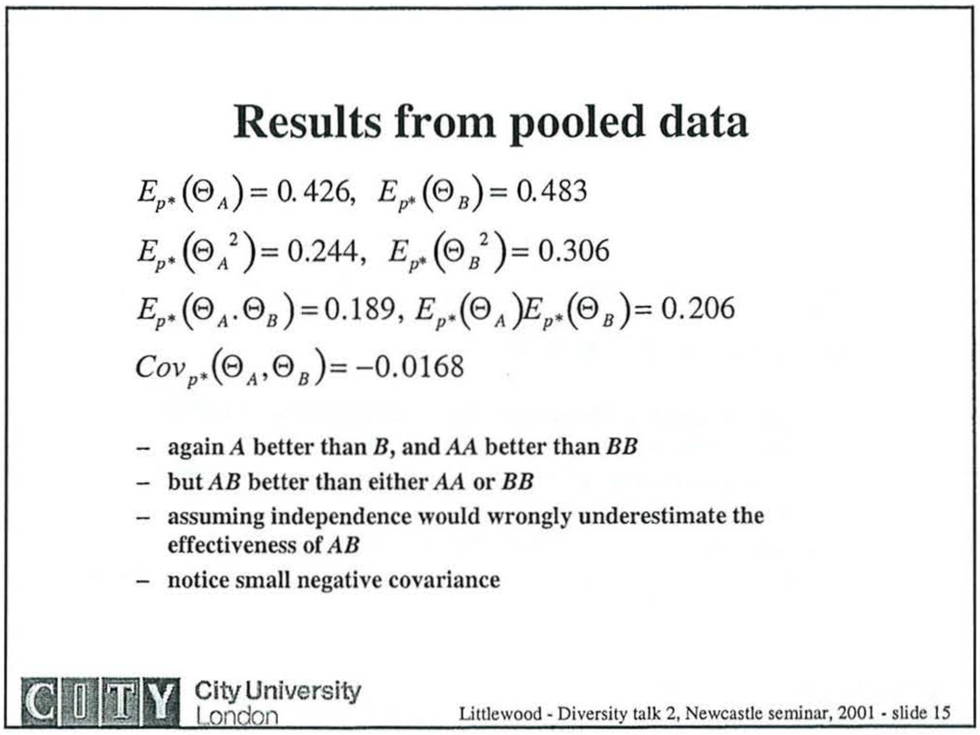
- applying *A* and *B* is better than 2*A*s or 2*B*s (expect only 17.9% of faults to remain, rather than 25.3%, 36.9% respectively)
- this is true even though *A* is better than *B* (and, more importantly, *AA* is better than *BB*)
- assuming *independence* underestimates *AB* efficacy
- note negative covariance here suggests that the procedures are 'quite diverse'

City University London

Comments

- Of course, this is an after-the-event analysis of a single program
- We really need to be able to *predict* what would happen for a *new* program
- If we are prepared to believe that these two programs are typical examples (a sample) of a class of programs, we could pool the data from them to obtain estimates of effectiveness of procedures *for the class*
- These can then be used to advise on best practice for a *new* program from the class
  - this is the kind of reasoning used *much more informally* for existing software cost models

City University London



Results from pooled data

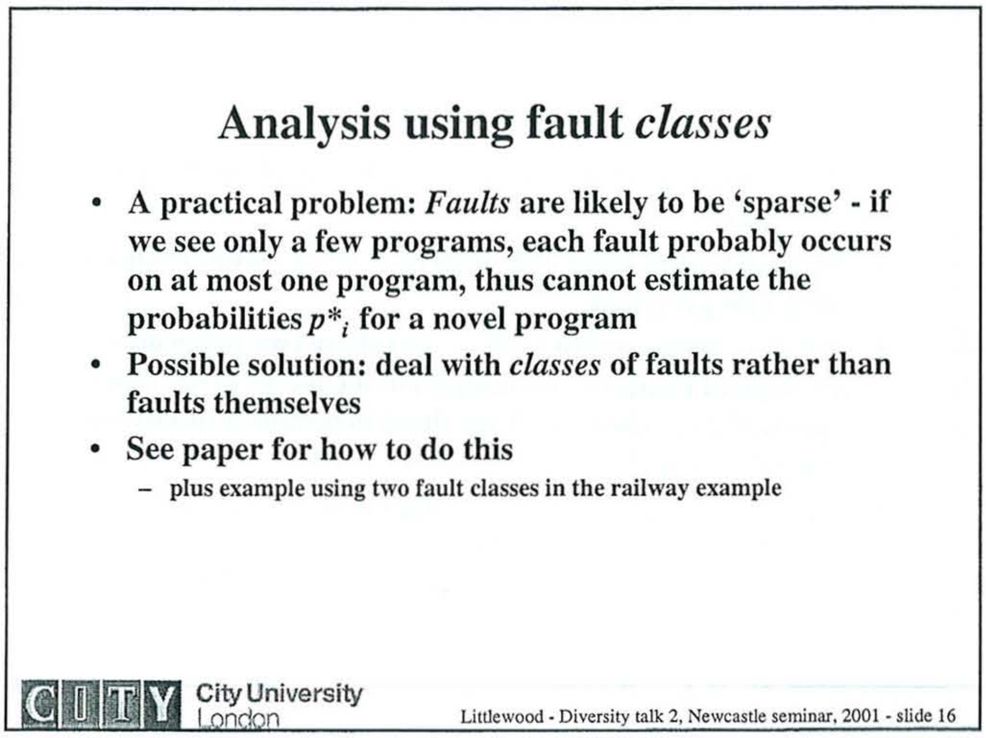
$E_{p*}(\Theta_A) = 0.426,\ E_{p*}(\Theta_B) = 0.483$

$E_{p*}(\Theta_A^{\ 2}) = 0.244,\ E_{p*}(\Theta_B^{\ 2}) = 0.306$

$E_{p*}(\Theta_A.\Theta_B) = 0.189,\ E_{p*}(\Theta_A)E_{p*}(\Theta_B) = 0.206$

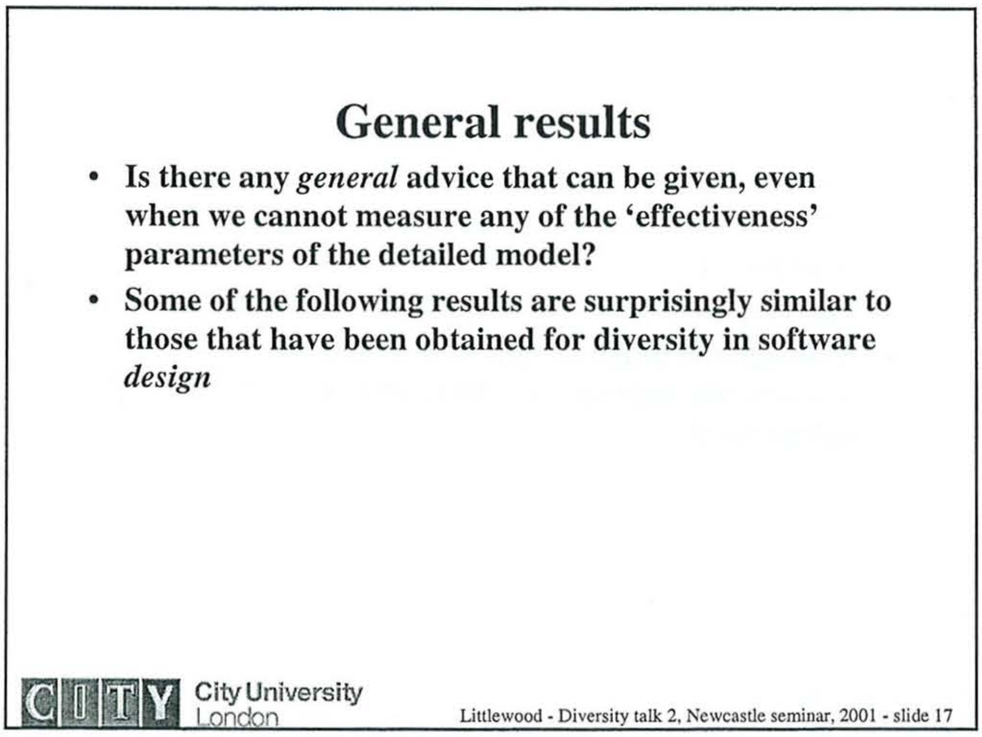
$Cov_{p*}(\Theta_A, \Theta_B) = -0.0168$

- again *A* better than *B*, and *AA* better than *BB*
- but *AB* better than either *AA* or *BB*
- assuming independence would wrongly underestimate the effectiveness of *AB*
- notice small negative covariance



Analysis using fault *classes*

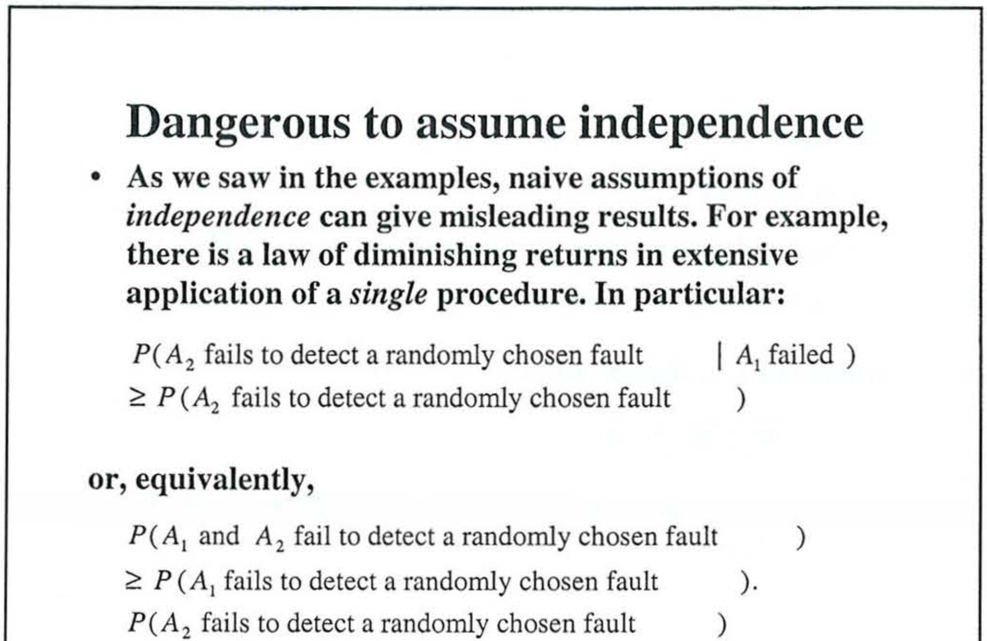
- A practical problem: *Faults* are likely to be 'sparse' - if we see only a few programs, each fault probably occurs on at most one program, thus cannot estimate the probabilities p^*_i for a novel program
- Possible solution: deal with *classes* of faults rather than faults themselves
- See paper for how to do this
  - plus example using two fault classes in the railway example



General results

- Is there any *general* advice that can be given, even when we cannot measure any of the 'effectiveness' parameters of the detailed model?
- Some of the following results are surprisingly similar to those that have been obtained for diversity in software *design*

City University London



Dangerous to assume independence

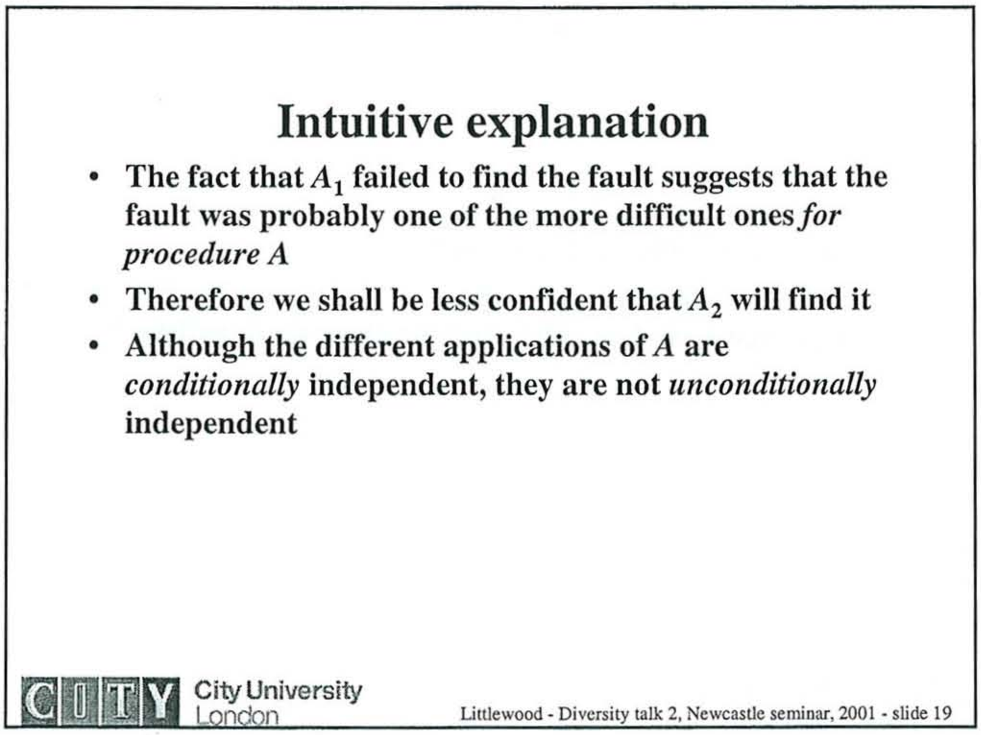
- As we saw in the examples, naive assumptions of *independence* can give misleading results. For example, there is a law of diminishing returns in extensive application of a *single* procedure. In particular:

$$P(A_2 \text{ fails to detect a randomly chosen fault} \mid A_1 \text{ failed})$$
$$\geq P(A_2 \text{ fails to detect a randomly chosen fault})$$

or, equivalently,

$$P(A_1 \text{ and } A_2 \text{ fail to detect a randomly chosen fault})$$
$$\geq P(A_1 \text{ fails to detect a randomly chosen fault}).$$
$$P(A_2 \text{ fails to detect a randomly chosen fault})$$

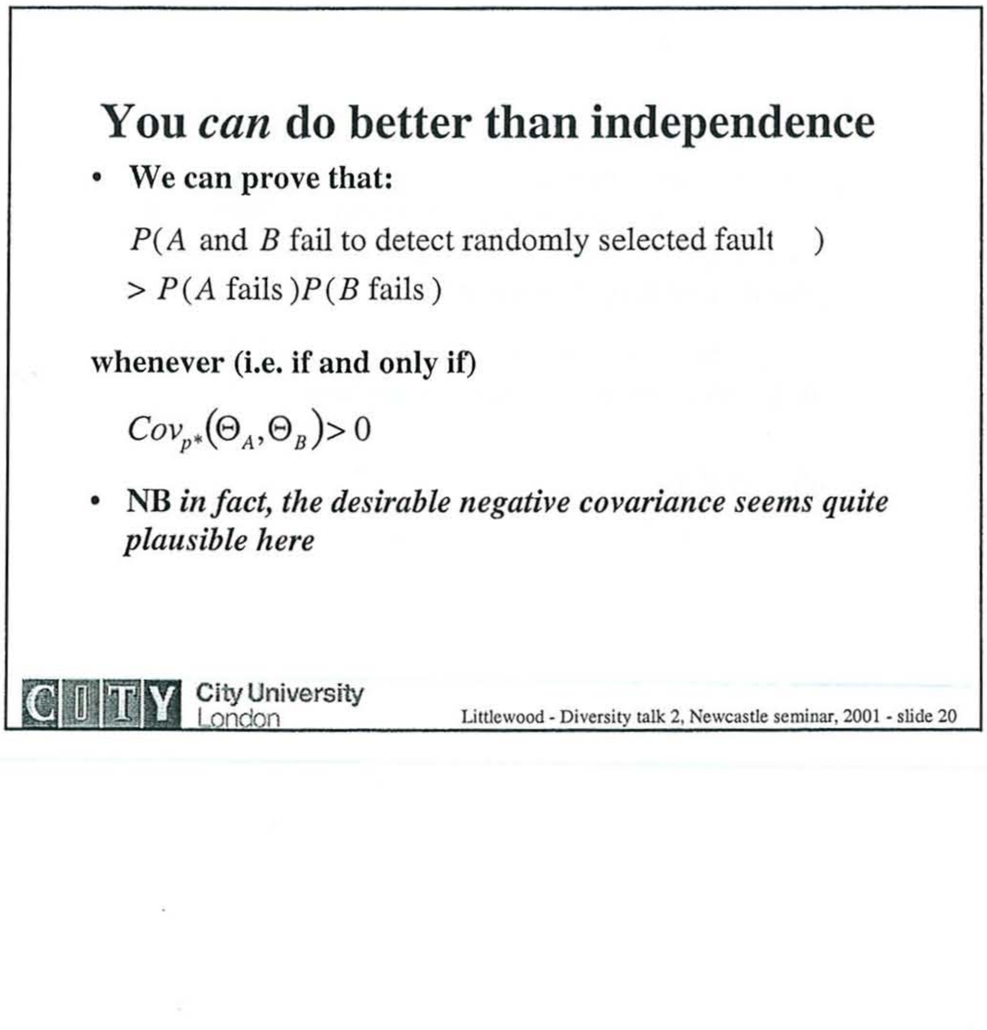
City University London



Intuitive explanation

- The fact that A_1 failed to find the fault suggests that the fault was probably one of the more difficult ones *for procedure A*
- Therefore we shall be less confident that A_2 will find it
- Although the different applications of A are *conditionally* independent, they are not *unconditionally* independent

City University London



You *can* do better than independence

- We can prove that:

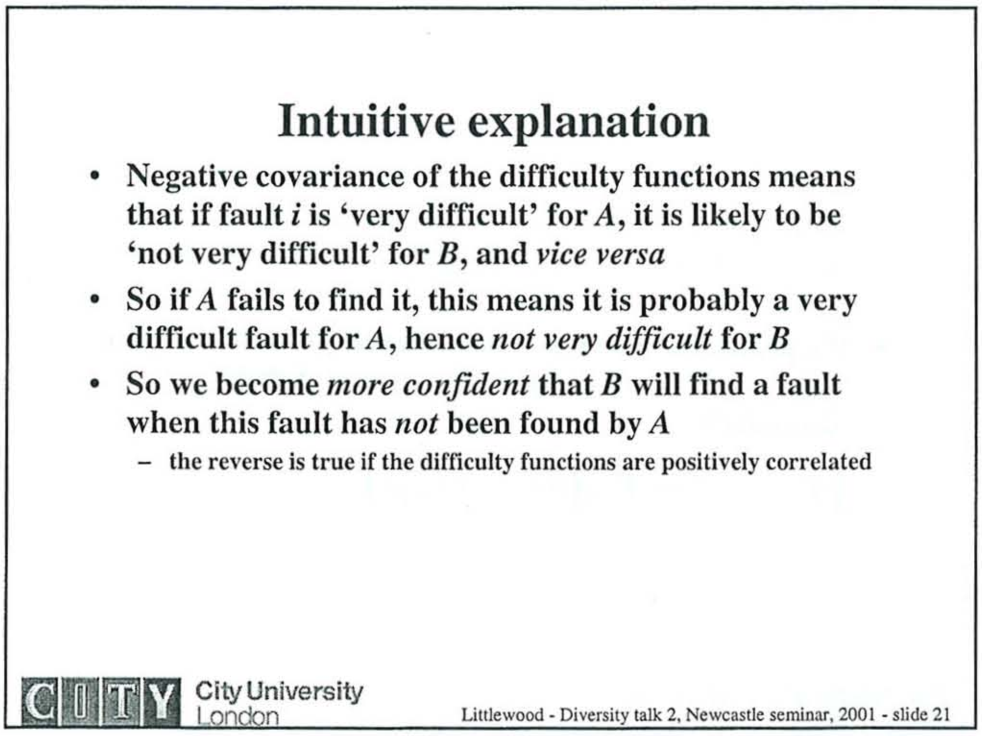
$$P(A \text{ and } B \text{ fail to detect randomly selected fault }) > P(A \text{ fails })P(B \text{ fails })$$

whenever (i.e. if and only if)

$$Cov_{p*}(\Theta_A, \Theta_B) > 0$$

- NB *in fact, the desirable negative covariance seems quite plausible here*

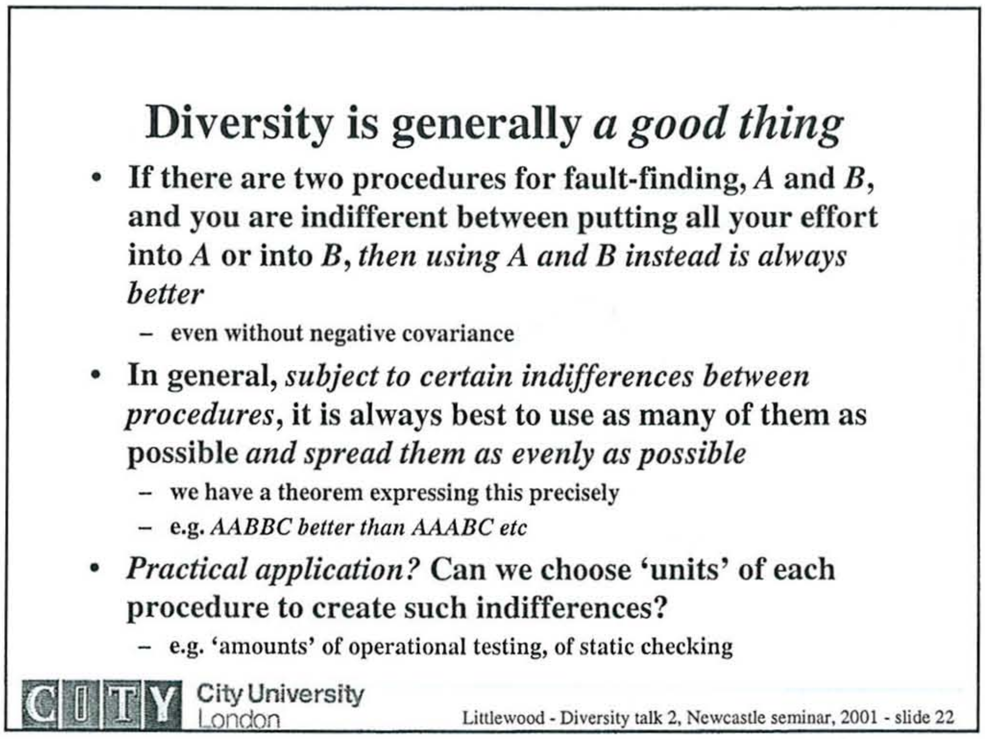
City University London



Intuitive explanation

- Negative covariance of the difficulty functions means that if fault *i* is 'very difficult' for *A*, it is likely to be 'not very difficult' for *B*, and *vice versa*
- So if *A* fails to find it, this means it is probably a very difficult fault for *A*, hence *not very difficult* for *B*
- So we become *more confident* that *B* will find a fault when this fault has *not* been found by *A*
  - the reverse is true if the difficulty functions are positively correlated

City University London



Diversity is generally *a good thing*

- If there are two procedures for fault-finding, *A* and *B*, and you are indifferent between putting all your effort into *A* or into *B*, *then using A and B instead is always better*
  - even without negative covariance
- In general, *subject to certain indifferences between procedures*, it is always best to use as many of them as possible *and spread them as evenly as possible*
  - we have a theorem expressing this precisely
  - e.g. *AABBC better than AAABC etc*
- *Practical application?* Can we choose 'units' of each procedure to create such indifferences?
  - e.g. 'amounts' of operational testing, of static checking

City University London

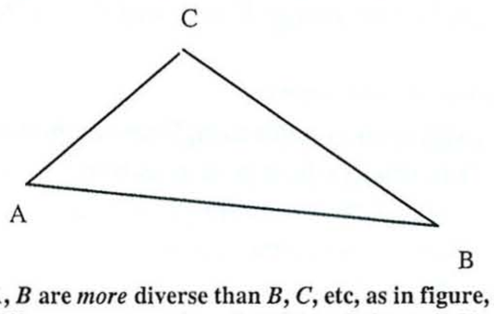
Diversity is generally *a good thing* (2)

- If diversity is a good thing, is it always better to try to get more of it? What do we mean by 'more'? Can we talk about *degrees* of diversity between fault-finding procedures?
- We propose a 'distance' between two procedures in the 'difficulty' function space - a measure of their 'diversity':

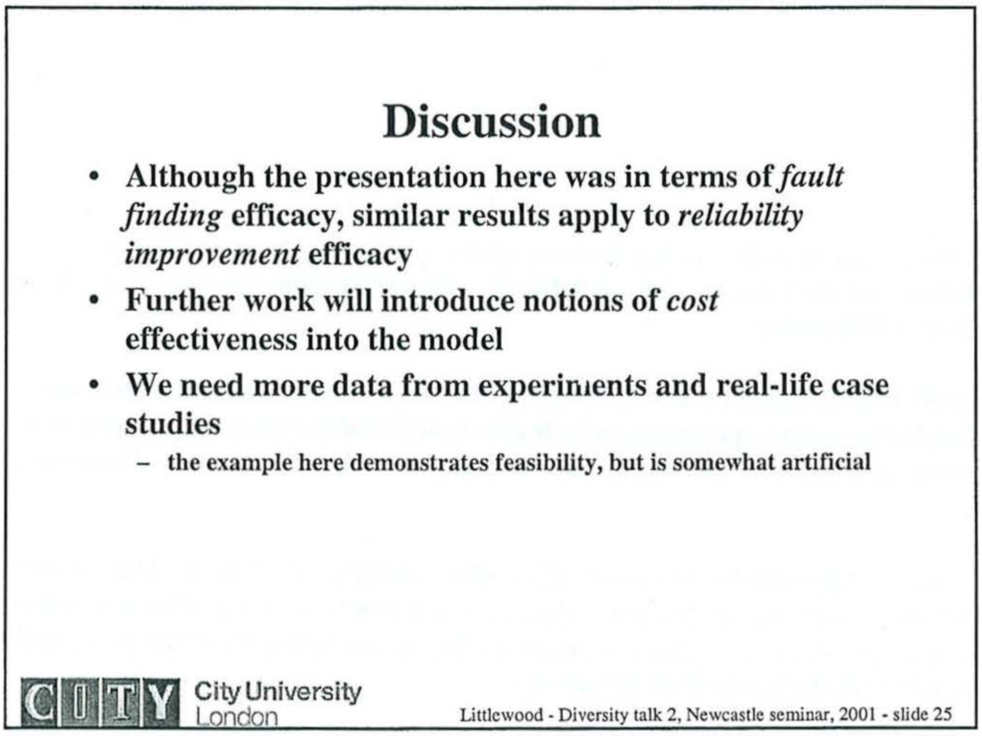
$$\|\theta_A - \theta_B\|^2 = E_{p*}\left((\Theta_A - \Theta_B)^2\right)$$

Diversity is generally *a good thing* (3)

- Subject to some 'indifference' assumptions, *more diversity* in this sense is better than *less*:



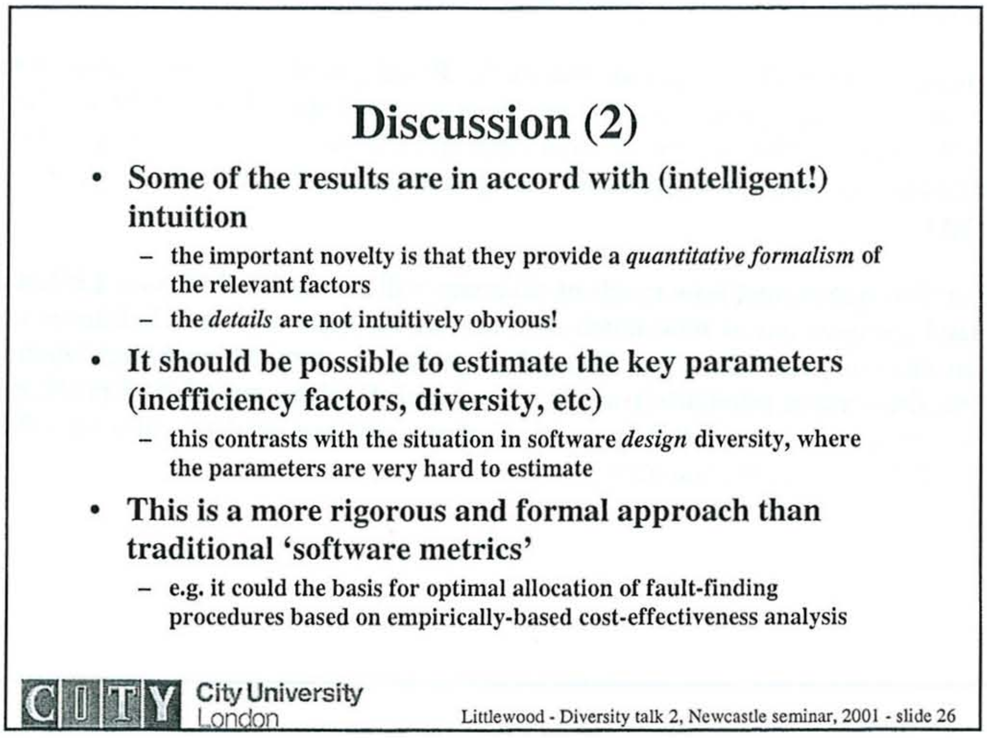
  - if *A*, *B* are *more* diverse than *B*, *C*, etc, as in figure, then subject to the indifference assumptions ('all things being equal') we would prefer *AB* to *BC*, and *BC* to *AC*



Discussion

- Although the presentation here was in terms of *fault finding* efficacy, similar results apply to *reliability improvement* efficacy
- Further work will introduce notions of *cost* effectiveness into the model
- We need more data from experiments and real-life case studies
  - the example here demonstrates feasibility, but is somewhat artificial

City University London



Discussion (2)

- Some of the results are in accord with (intelligent!) intuition
  - the important novelty is that they provide a *quantitative formalism* of the relevant factors
  - the *details* are not intuitively obvious!
- It should be possible to estimate the key parameters (inefficiency factors, diversity, etc)
  - this contrasts with the situation in software *design* diversity, where the parameters are very hard to estimate
- This is a more rigorous and formal approach than traditional 'software metrics'
  - e.g. it could the basis for optimal allocation of fault-finding procedures based on empirically-based cost-effectiveness analysis

City University London

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.

DISCUSSION

Rapporteur: Dr L.B. Arief

Lecture Two

Dr Maxion wanted to know whether the variation in the difficulty function is bimodal. Professor Littlewood replied that he does not know for sure, but there will be some statistical data available.

Dr Horning asked for a clarification on the difficulty function, whether it concerns fault 'i' or fault 'type i'. Professor Littlewood answered that (for now) it is fault 'i', not fault 'type i'

Dr Stroud questioned the efficiency of running the same procedure multiple times. Professor Littlewood argued that it is possible to catch fresh bugs in the later runs, although there is a law of diminishing returns.

Professor Schneider was not sure about the application of the technique in reality and he wondered whether it can accurately model actual software engineering techniques. Professor Littlewood replied that he would show some fairly artificial examples, where two actual software engineering procedures can be estimated using the technique, which proves to be useful in giving quantitative insight into what is happening.

Professor Malek questioned the efficiency of having two procedures, where one takes a lot longer than the other to achieve the same detection. Professor Littlewood replied that issues like these would be addressed later in the talk. Professor Littlewood wished to eventually come down to an indifference notion where they both cost the same and be equally effective. So, the concerns would be on whether to use both or two of each.

Dr Lomet wondered whether there are some assumptions on independence of the ability to detect faults. Professor Littlewood said that he is assuming independence to be conditional. Dr Ross then asked whether the two procedures commute. Professor Littlewood replied that we really do not care about that, and the conditional independence is quite reasonable. Regarding multiple procedures, Mr Mpoeleng asked whether fault correction is assumed after the first procedure. Professor Littlewood pointed out that no correction is performed, we are only interested in the most effective way of finding faults.

Dr Stroud wanted to confirm that the numbers shown are something that Professor Littlewood calculated, not what the students did in the experiment. This was confirmed by Professor Littlewood, and was then re-affirmed by Professor Anderson. He mentioned that the data came from the work that the students did in the experiment and Professor Littlewood did the calculation using those data to come up with the numbers.

Dr Stroud then asked whether deploying it twice means that the same students did it twice, or different students did it each time, or average students did it twice. Professor Littlewood replied that in the case of testing, there would be an operational test generator, and doing it

Dr Lomet argued that people usually find bugs, fix them, re-test, and find some more bugs. So in some cases, there are bugs in a program which could not be found in the first collection of tests because they are masked by the previous bugs. Professor Littlewood replied that the model does not consider that. Dr Lomet then said that in real world, the probabilities do change every time the procedure is run because the previous bugs that have been found were fixed before the procedure is re-applied. Professor Littlewood admitted that the model does not cope with that kind of situation.

Dr Horning suggested two experiments. The first one takes procedure A and re-labels it A'. Professor Littlewood said he would treat them as the same and he did not see a problem with that. Professor Jones mentioned the triangle theory as an answer, where we would want ones with as different difficulty functions as possible. The second suggestion concerns an experiment where there is an error that could never be found. Professor Littlewood replied that it might happen, since the difficulty function can have a value between zero and one, and when it is one, it means that the bug will never be found.

Dr Maxion suggested that some of the difficulties might be resolved using details of psychological experiments. Professor Littlewood acknowledged that introducing another person can be a new form of diversity (human diversity). Dr Maxion also commented on interdisciplinary work, that there is not a perfect understanding in each other's area. Professor Littlewood replied that the psychologists he is working with are working with software engineers, and they are funded by Railtrack. There is a strong interest in safety critical signalling systems, in particular in finding faults in people who have written signalling software. The psychologists had special psychological hypotheses in mind, but they were working with people who really understood the railway problems.