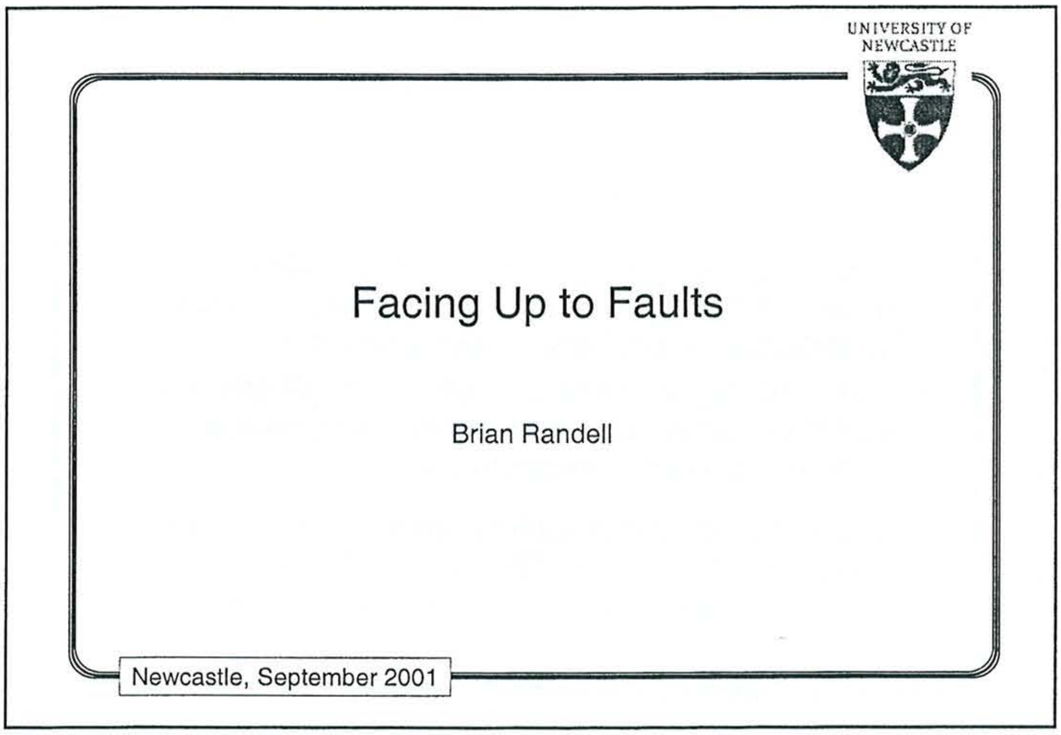
V

FACING UP TO FAULTS

B Randell

Rapporteur: Professor M Koutny

V.2

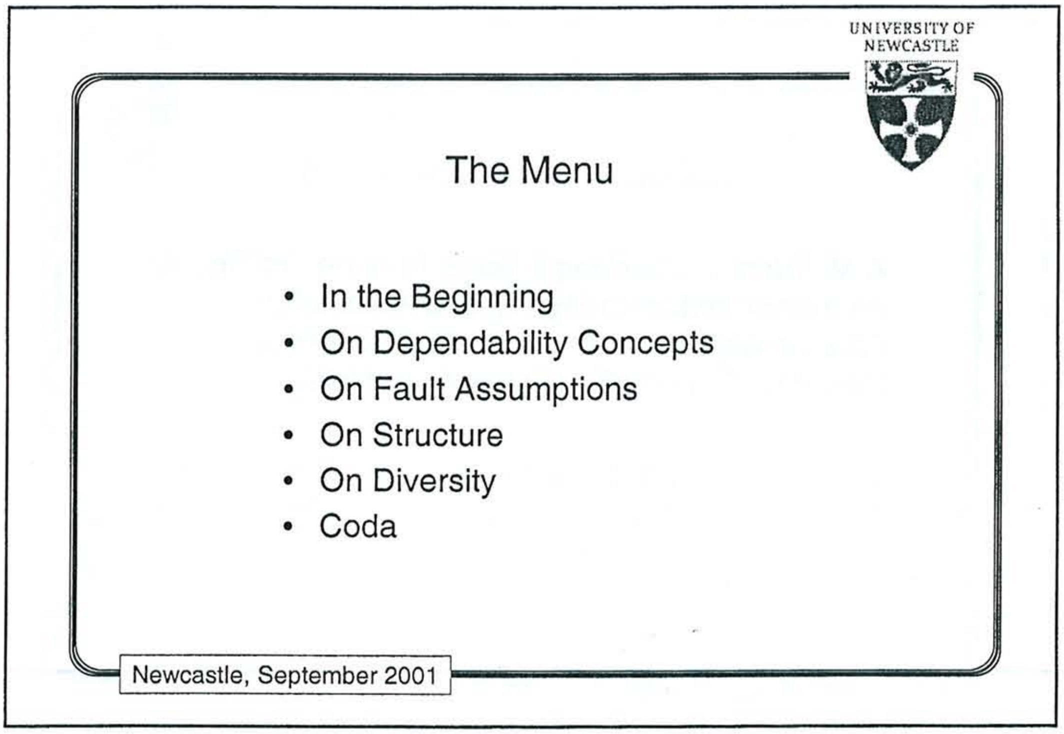


UNIVERSITY OF NEWCASTLE

Facing Up to Faults

Brian Randell

Newcastle, September 2001

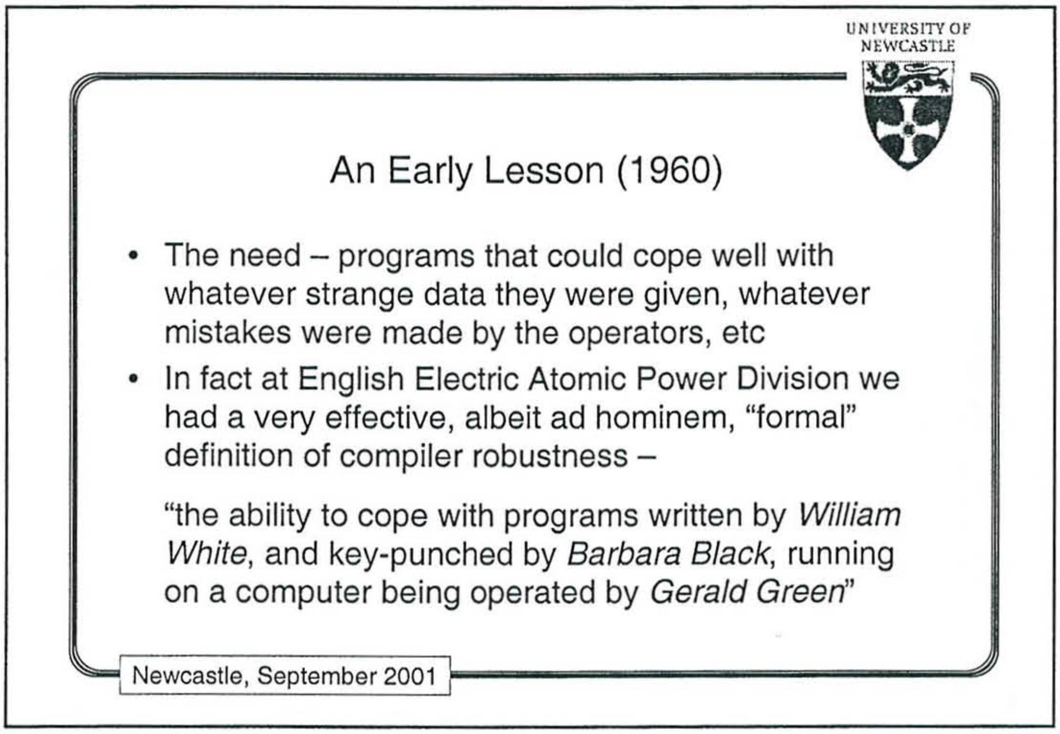


UNIVERSITY OF NEWCASTLE

The Menu

- In the Beginning
- On Dependability Concepts
- On Fault Assumptions
- On Structure
- On Diversity
- Coda

Newcastle, September 2001



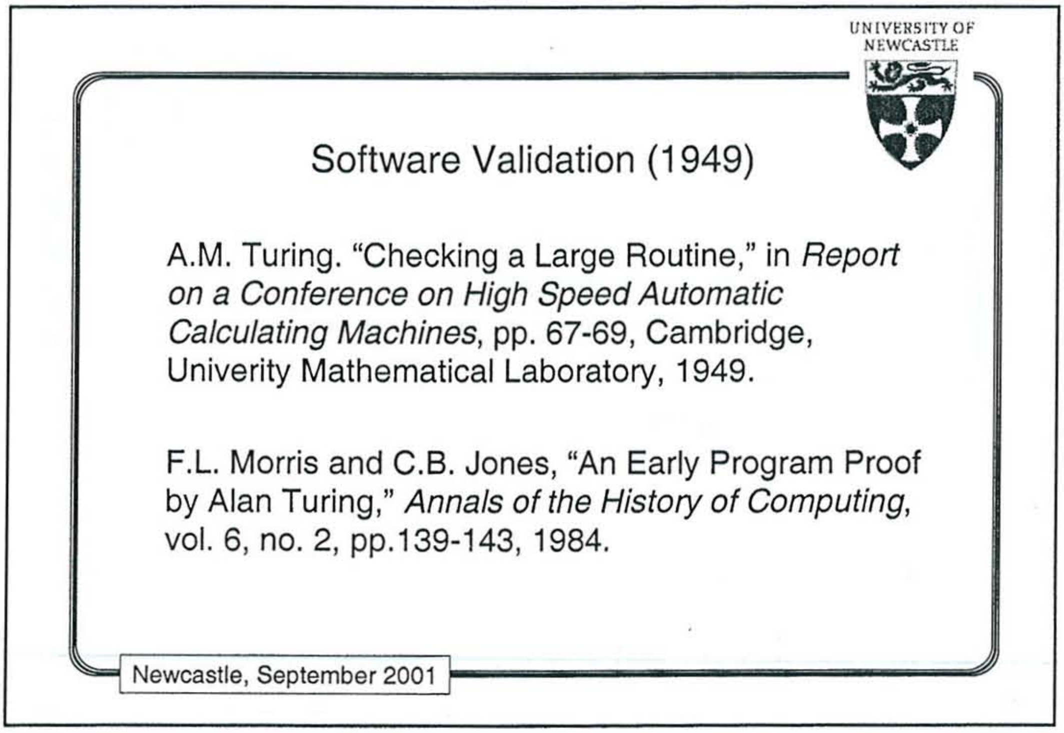
UNIVERSITY OF NEWCASTLE

An Early Lesson (1960)

- The need – programs that could cope well with whatever strange data they were given, whatever mistakes were made by the operators, etc
- In fact at English Electric Atomic Power Division we had a very effective, albeit ad hominem, "formal" definition of compiler robustness –

"the ability to cope with programs written by *William White*, and key-punched by *Barbara Black*, running on a computer being operated by *Gerald Green*"

Newcastle, September 2001



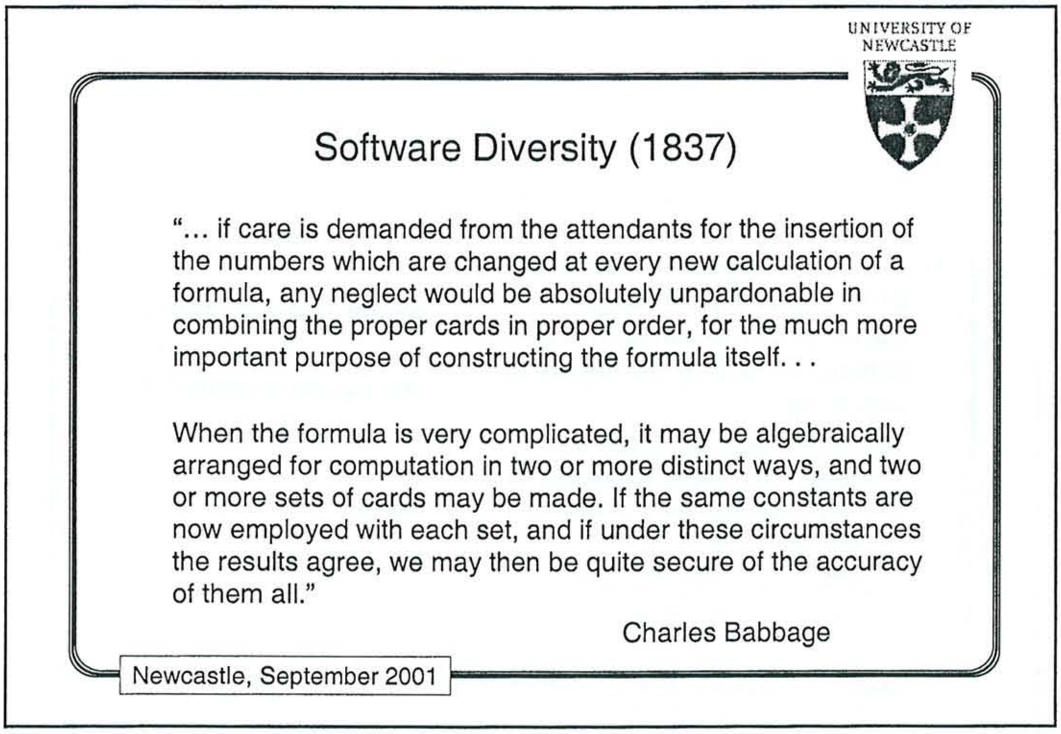
UNIVERSITY OF NEWCASTLE

Software Validation (1949)

A.M. Turing. "Checking a Large Routine," in *Report on a Conference on High Speed Automatic Calculating Machines*, pp. 67-69, Cambridge, Univerity Mathematical Laboratory, 1949.

F.L. Morris and C.B. Jones, "An Early Program Proof by Alan Turing," *Annals of the History of Computing*, vol. 6, no. 2, pp.139-143, 1984.

Newcastle, September 2001



UNIVERSITY OF NEWCASTLE

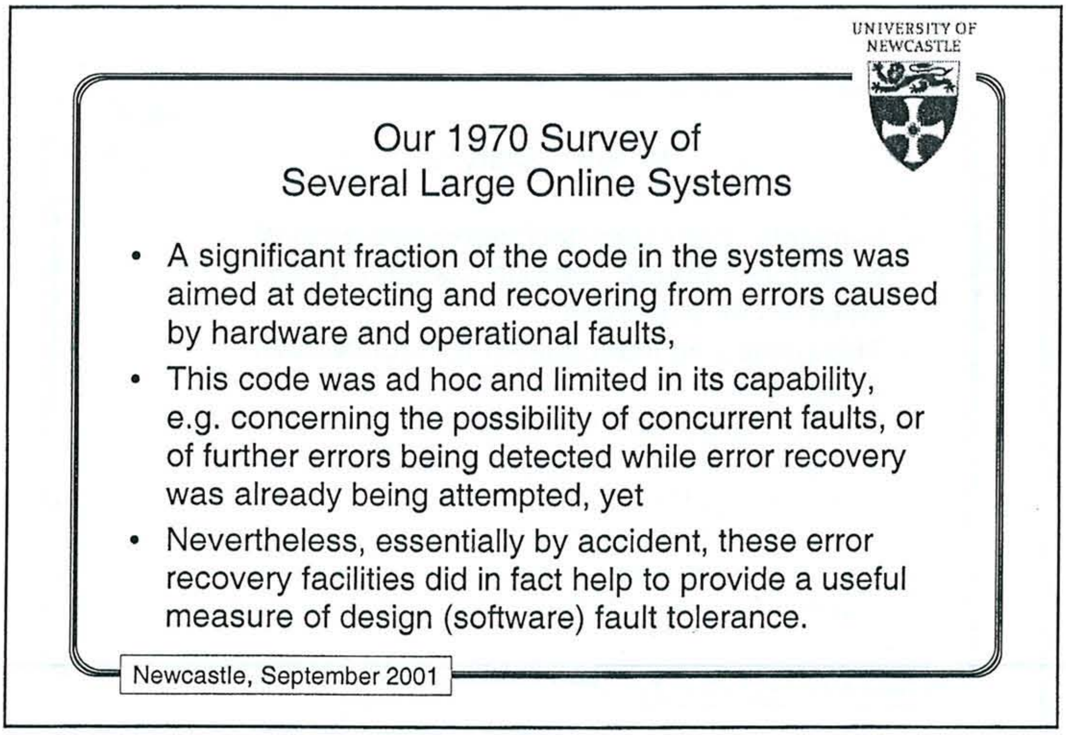
Software Diversity (1837)

"... if care is demanded from the attendants for the insertion of the numbers which are changed at every new calculation of a formula, any neglect would be absolutely unpardonable in combining the proper cards in proper order, for the much more important purpose of constructing the formula itself. . .

When the formula is very complicated, it may be algebraically arranged for computation in two or more distinct ways, and two or more sets of cards may be made. If the same constants are now employed with each set, and if under these circumstances the results agree, we may then be quite secure of the accuracy of them all."

Charles Babbage

Newcastle, September 2001

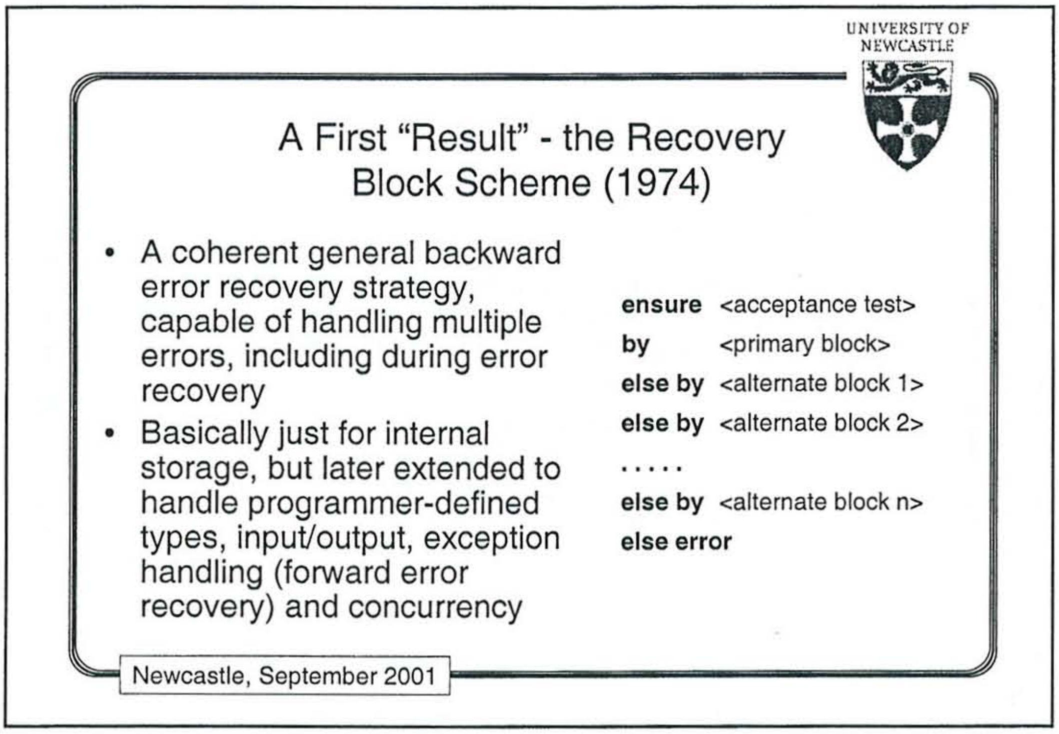


UNIVERSITY OF NEWCASTLE

Our 1970 Survey of Several Large Online Systems

- A significant fraction of the code in the systems was aimed at detecting and recovering from errors caused by hardware and operational faults,
- This code was ad hoc and limited in its capability, e.g. concerning the possibility of concurrent faults, or of further errors being detected while error recovery was already being attempted, yet
- Nevertheless, essentially by accident, these error recovery facilities did in fact help to provide a useful measure of design (software) fault tolerance.

Newcastle, September 2001



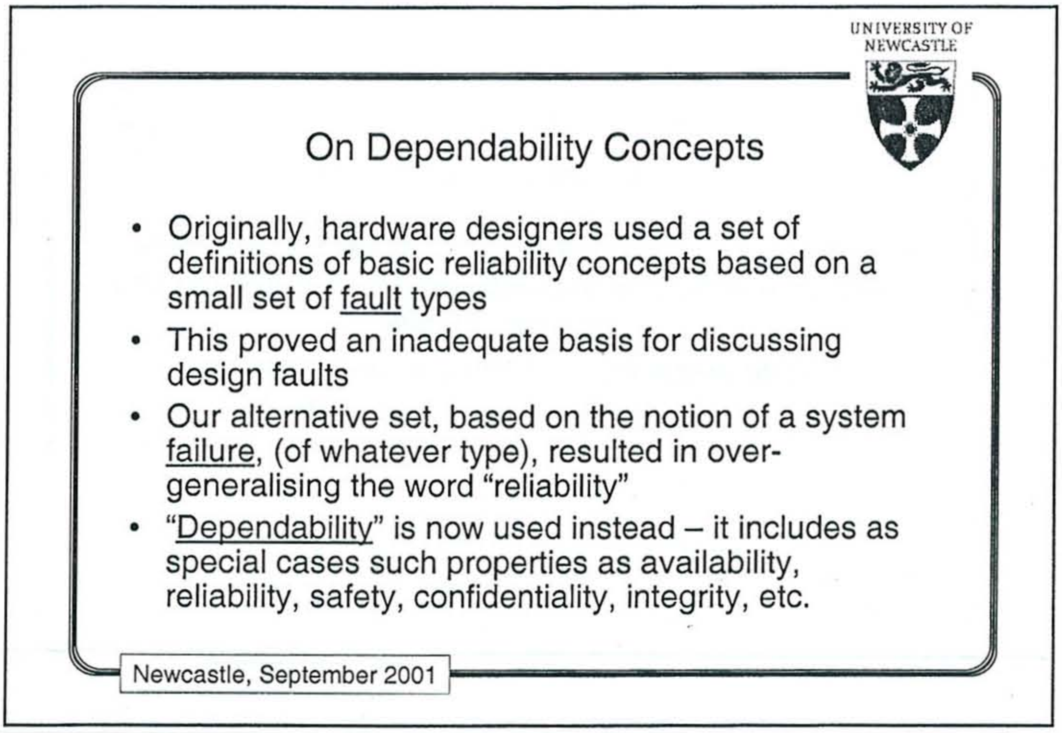
UNIVERSITY OF NEWCASTLE

A First "Result" - the Recovery Block Scheme (1974)

- A coherent general backward error recovery strategy, capable of handling multiple errors, including during error recovery
- Basically just for internal storage, but later extended to handle programmer-defined types, input/output, exception handling (forward error recovery) and concurrency

```
ensure    <acceptance test>
by        <primary block>
else by   <alternate block 1>
else by   <alternate block 2>
.....
else by   <alternate block n>
else error
```

Newcastle, September 2001

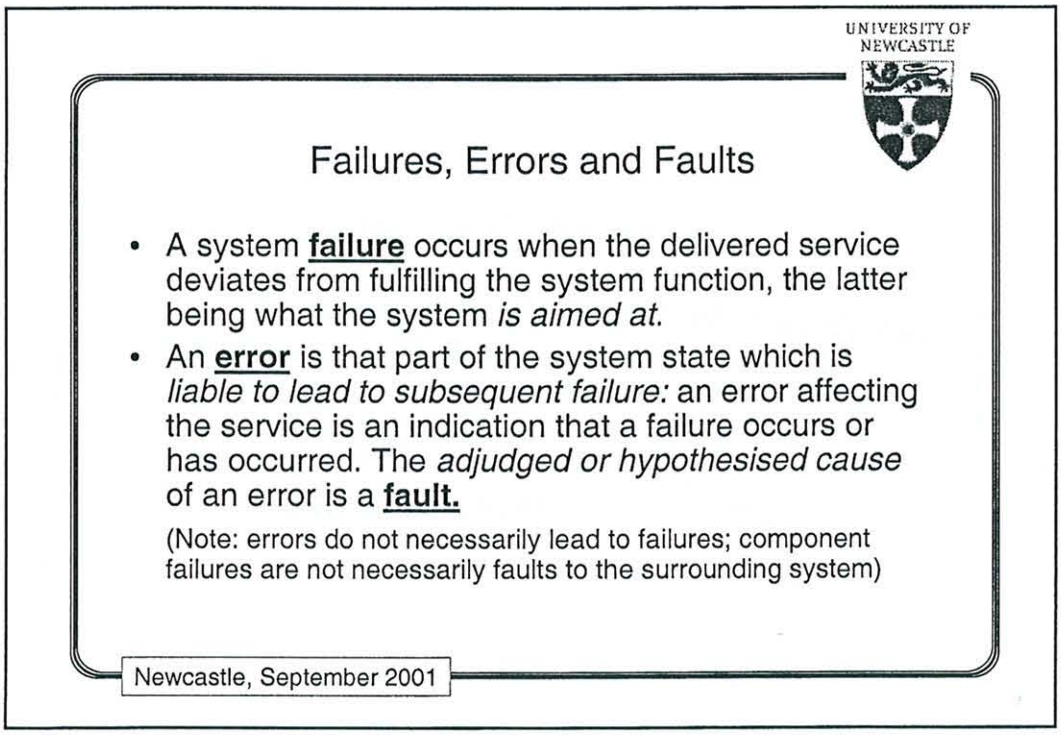


UNIVERSITY OF NEWCASTLE

On Dependability Concepts

- Originally, hardware designers used a set of definitions of basic reliability concepts based on a small set of fault types
- This proved an inadequate basis for discussing design faults
- Our alternative set, based on the notion of a system failure, (of whatever type), resulted in over-generalising the word "reliability"
- "Dependability" is now used instead – it includes as special cases such properties as availability, reliability, safety, confidentiality, integrity, etc.

Newcastle, September 2001



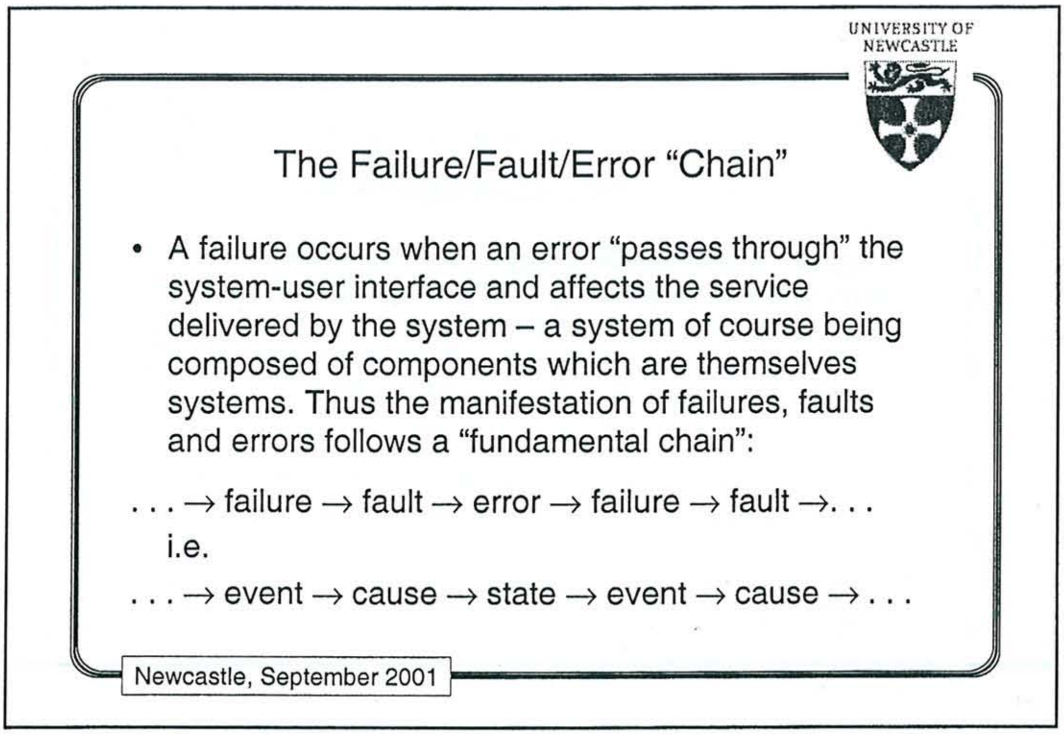
UNIVERSITY OF NEWCASTLE

Failures, Errors and Faults

- A system **failure** occurs when the delivered service deviates from fulfilling the system function, the latter being what the system *is aimed at.*
- An **error** is that part of the system state which is *liable to lead to subsequent failure:* an error affecting the service is an indication that a failure occurs or has occurred. The *adjudged or hypothesised cause* of an error is a **fault.**

  (Note: errors do not necessarily lead to failures; component failures are not necessarily faults to the surrounding system)

Newcastle, September 2001



UNIVERSITY OF NEWCASTLE

The Failure/Fault/Error "Chain"

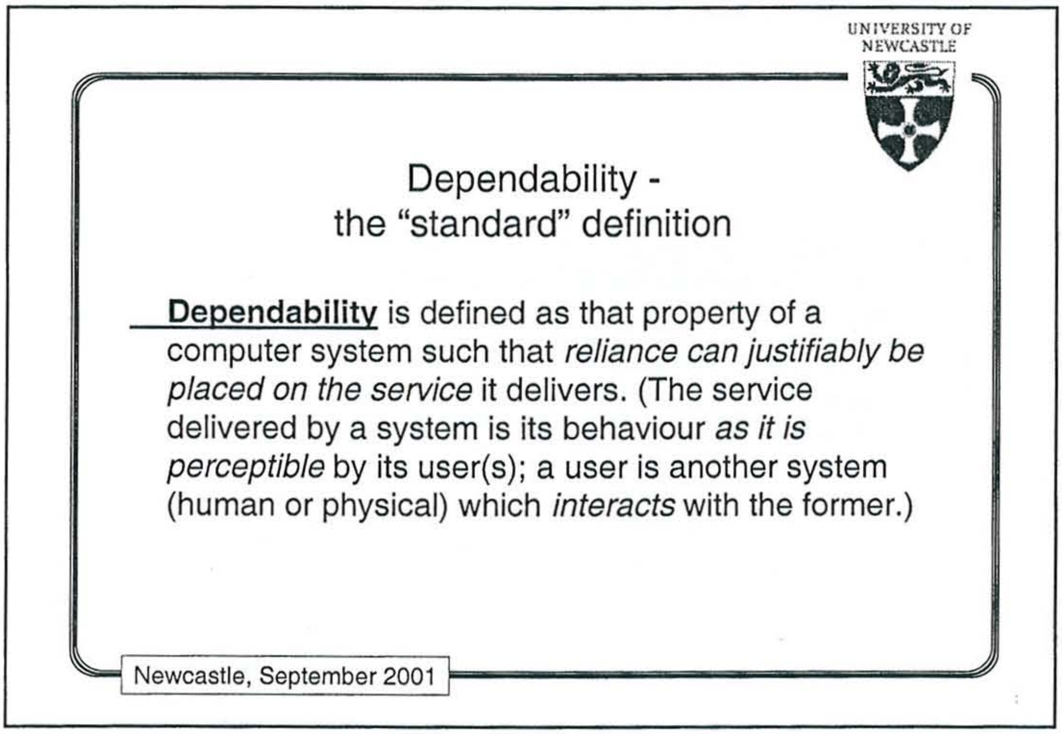
- A failure occurs when an error "passes through" the system-user interface and affects the service delivered by the system – a system of course being composed of components which are themselves systems. Thus the manifestation of failures, faults and errors follows a "fundamental chain":

. . . → failure → fault → error → failure → fault →. . .

i.e.

. . . → event → cause → state → event → cause → . . .

Newcastle, September 2001

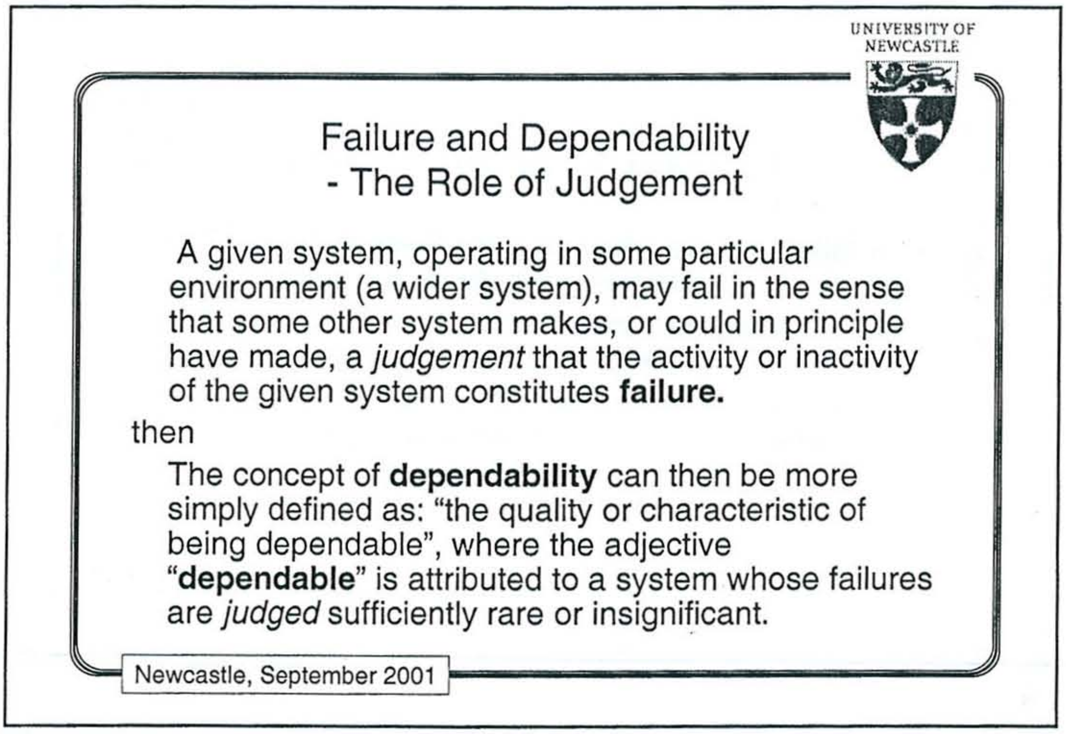


UNIVERSITY OF NEWCASTLE

Dependability - the "standard" definition

Dependability is defined as that property of a computer system such that *reliance can justifiably be placed on the service* it delivers. (The service delivered by a system is its behaviour *as it is perceptible* by its user(s); a user is another system (human or physical) which *interacts* with the former.)

Newcastle, September 2001



UNIVERSITY OF NEWCASTLE

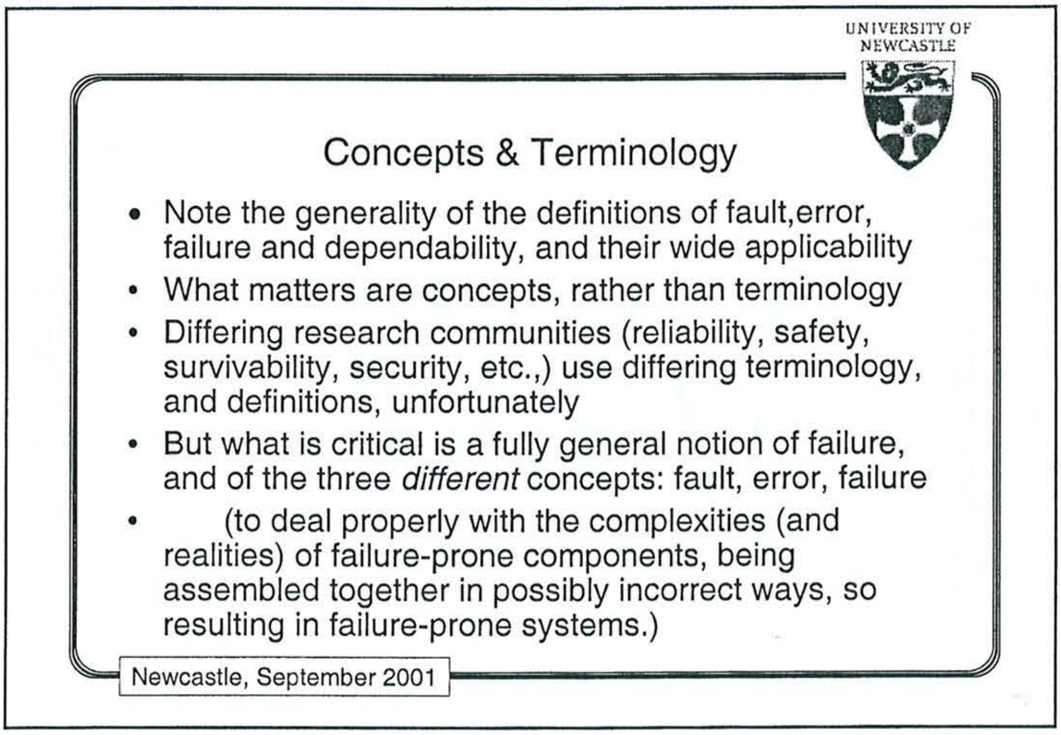
Failure and Dependability - The Role of Judgement

A given system, operating in some particular environment (a wider system), may fail in the sense that some other system makes, or could in principle have made, a *judgement* that the activity or inactivity of the given system constitutes **failure.**

then

The concept of **dependability** can then be more simply defined as: "the quality or characteristic of being dependable", where the adjective "**dependable**" is attributed to a system whose failures are *judged* sufficiently rare or insignificant.

Newcastle, September 2001

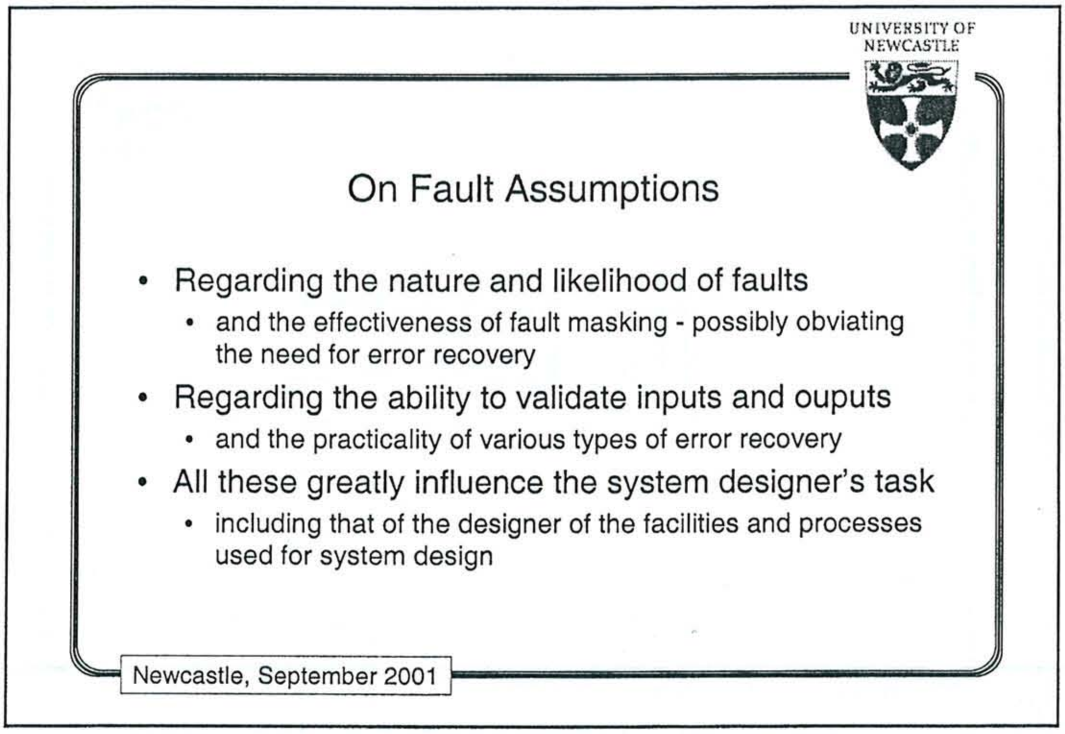


UNIVERSITY OF NEWCASTLE

Concepts & Terminology

- Note the generality of the definitions of fault,error, failure and dependability, and their wide applicability
- What matters are concepts, rather than terminology
- Differing research communities (reliability, safety, survivability, security, etc.,) use differing terminology, and definitions, unfortunately
- But what is critical is a fully general notion of failure, and of the three *different* concepts: fault, error, failure
- (to deal properly with the complexities (and realities) of failure-prone components, being assembled together in possibly incorrect ways, so resulting in failure-prone systems.)

Newcastle, September 2001

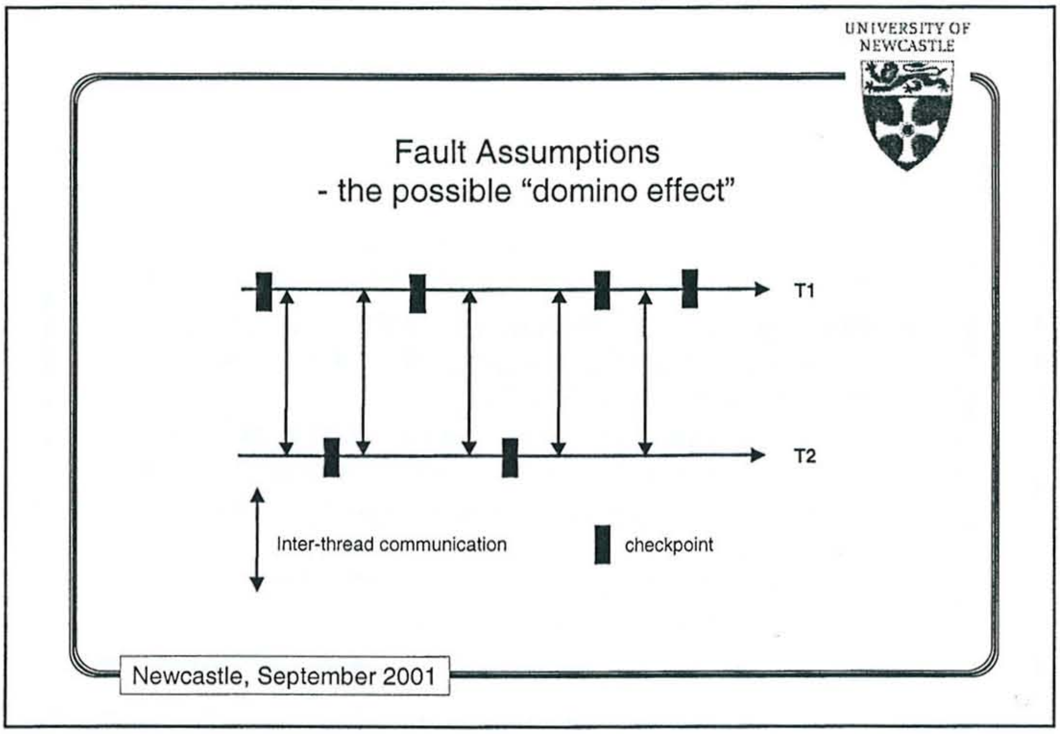


UNIVERSITY OF NEWCASTLE

On Fault Assumptions

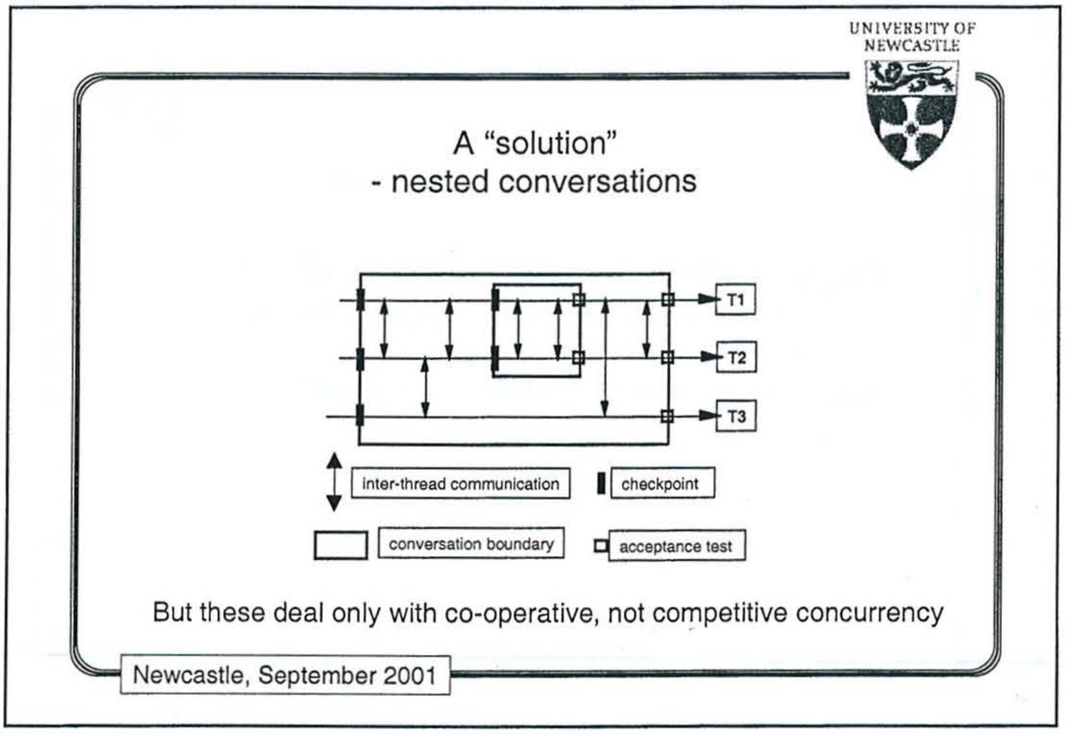
- Regarding the nature and likelihood of faults
  - and the effectiveness of fault masking - possibly obviating the need for error recovery
- Regarding the ability to validate inputs and ouputs
  - and the practicality of various types of error recovery
- All these greatly influence the system designer's task
  - including that of the designer of the facilities and processes used for system design

Newcastle, September 2001



Fault Assumptions - the possible "domino effect"

Newcastle, September 2001



A "solution" - nested conversations

But these deal only with co-operative, not competitive concurrency

Newcastle, September 2001

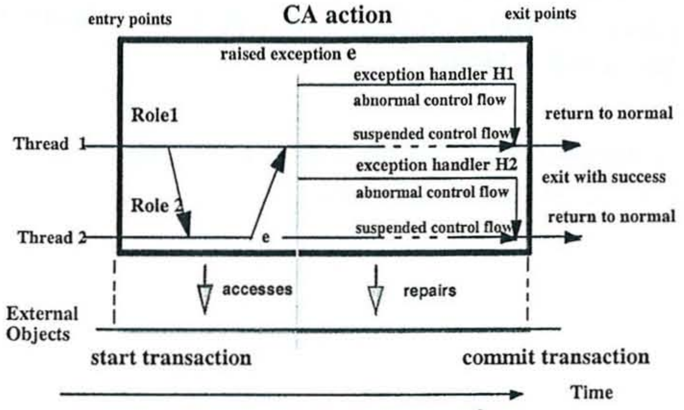
UNIVERSITY OF NEWCASTLE

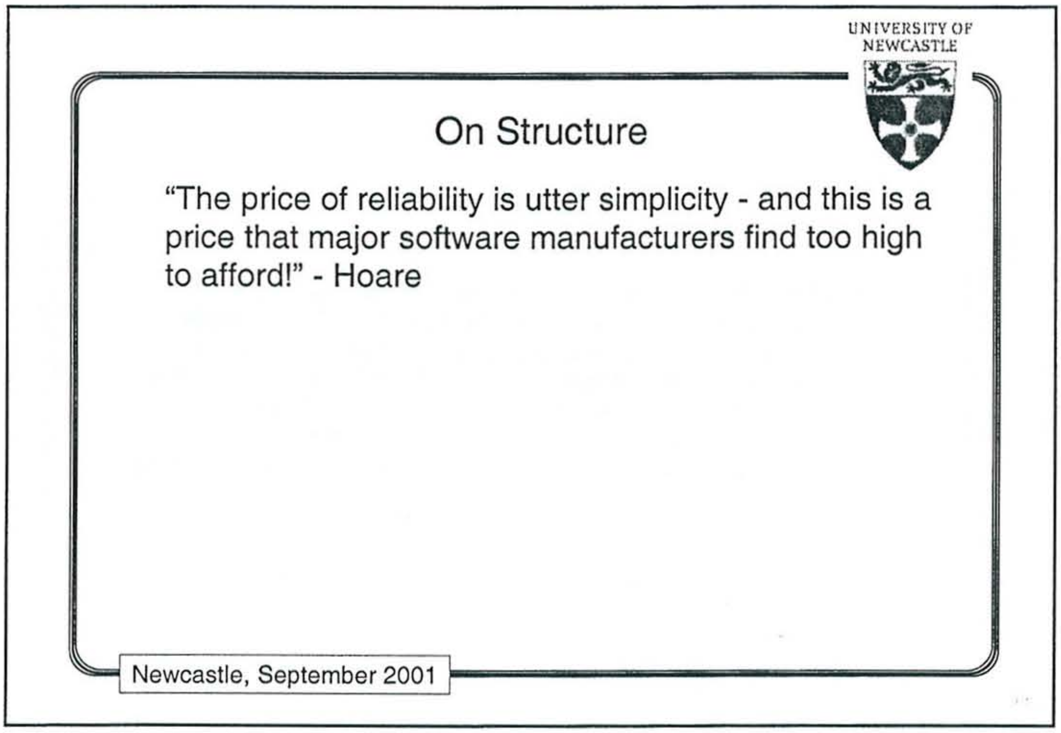
Co-ordinated Atomic Actions

- A mechanism/protocol for (forward and/or backward) error recovery for systems and their environments in the presence of both cooperative and competitive concurrency.
- In effect a programming discipline for nested multi-threaded transactions with very general exception handling provisions
  - To cooperate in a CA action a group of concurrent threads must come together to perform the roles of the action collectively. They enter and leave the action in real or virtual <u>synchrony</u>
  - Inside a CA action, roles can be involved in (nested CA actions.
  - If an error is detected inside a CA action, recovery measures must be invoked co-operatively, by <u>all</u> the roles, in order to reach some mutually consistent conclusion (success, exception, or failure)
  - External objects, which are in effect being competed for by the CA action, must behave <u>atomically</u> with respect to other CA actions and threads so that they cannot be used as an implicit means of "smuggling" information into or out of a CA action.

UNIVERSITY OF NEWCASTLE

A Co-ordinated Atomic Action



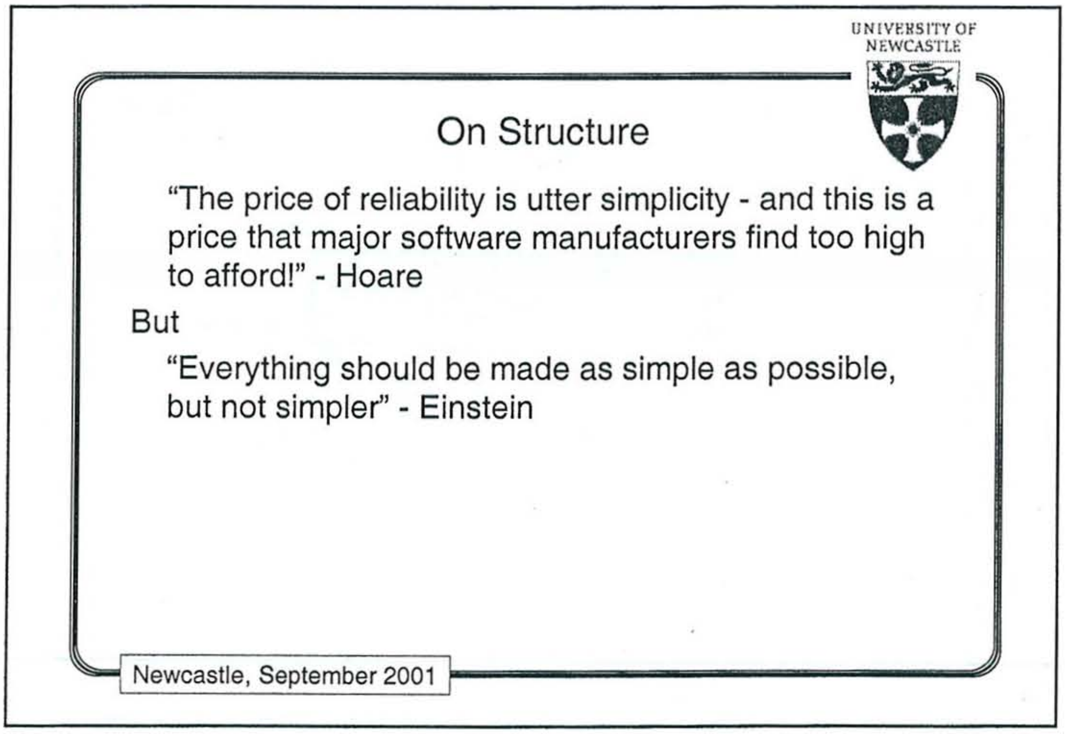


UNIVERSITY OF NEWCASTLE

On Structure

"The price of reliability is utter simplicity - and this is a price that major software manufacturers find too high to afford!" - Hoare

Newcastle, September 2001



UNIVERSITY OF NEWCASTLE

On Structure

"The price of reliability is utter simplicity - and this is a price that major software manufacturers find too high to afford!" - Hoare

But

"Everything should be made as simple as possible, but not simpler" - Einstein

Newcastle, September 2001

UNIVERSITY OF NEWCASTLE

On Structure

"The price of reliability is utter simplicity - and this is a price that major software manufacturers find too high to afford!" - Hoare

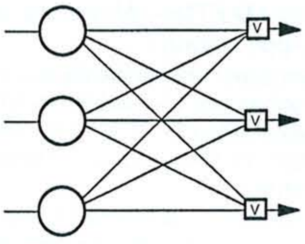
But

"Everything should be made as simple as possible, but not simpler" - Einstein

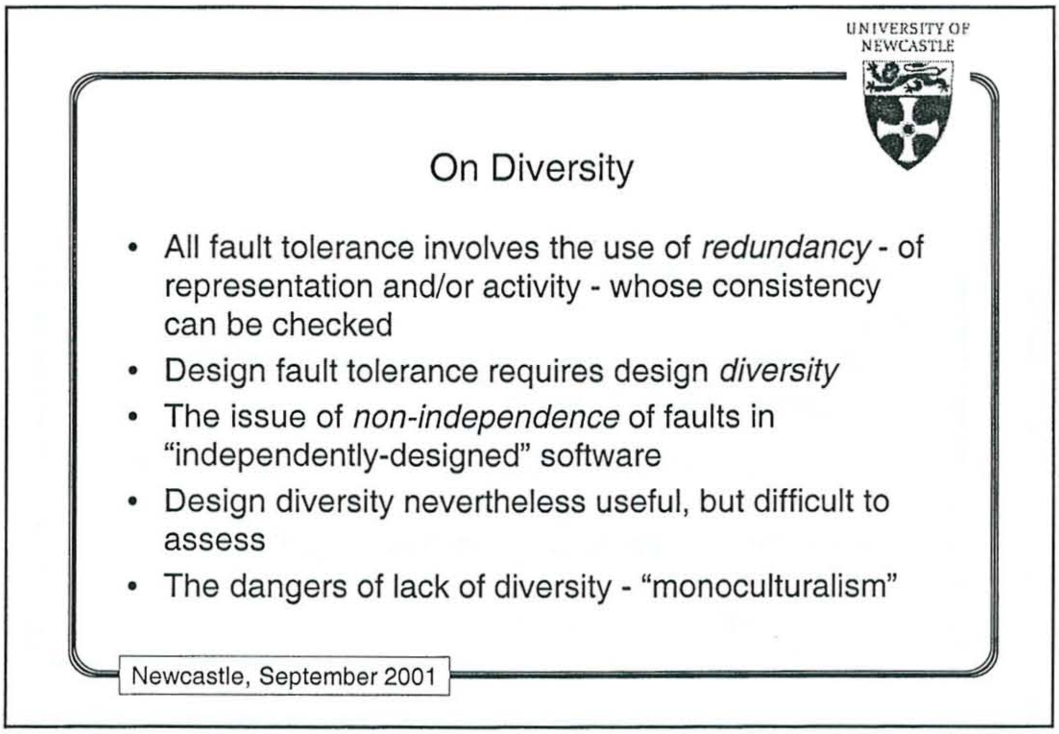
- Good system structuring allows one to deal with the added complexity that result from more realistic fault assumptions - its quality is measured by its:
  - coupling and cohesion (for performance)
  - **strength** (for dependability)

UNIVERSITY OF NEWCASTLE

An Example - Triple Modular Redundancy



A strongly-structured system is one in which the structuring exists in the actual system, not just its description or design, and helps to limit the impact of faults

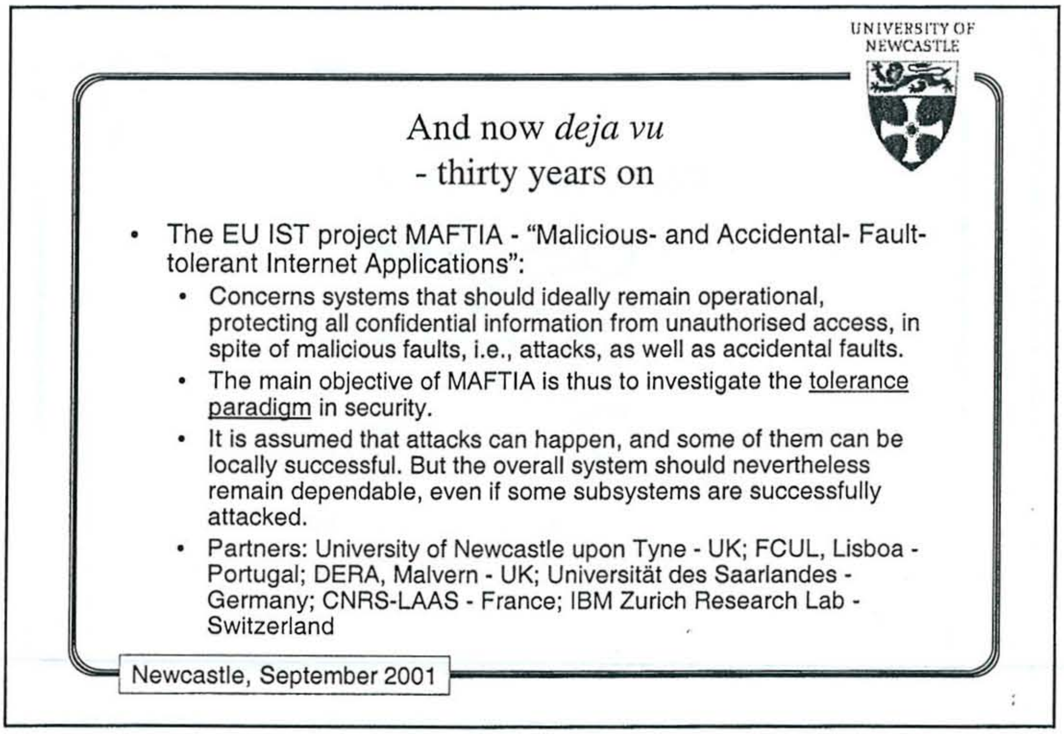


UNIVERSITY OF NEWCASTLE

On Diversity

- All fault tolerance involves the use of *redundancy* - of representation and/or activity - whose consistency can be checked
- Design fault tolerance requires design *diversity*
- The issue of *non-independence* of faults in “independently-designed” software
- Design diversity nevertheless useful, but difficult to assess
- The dangers of lack of diversity - “monoculturalism”

Newcastle, September 2001

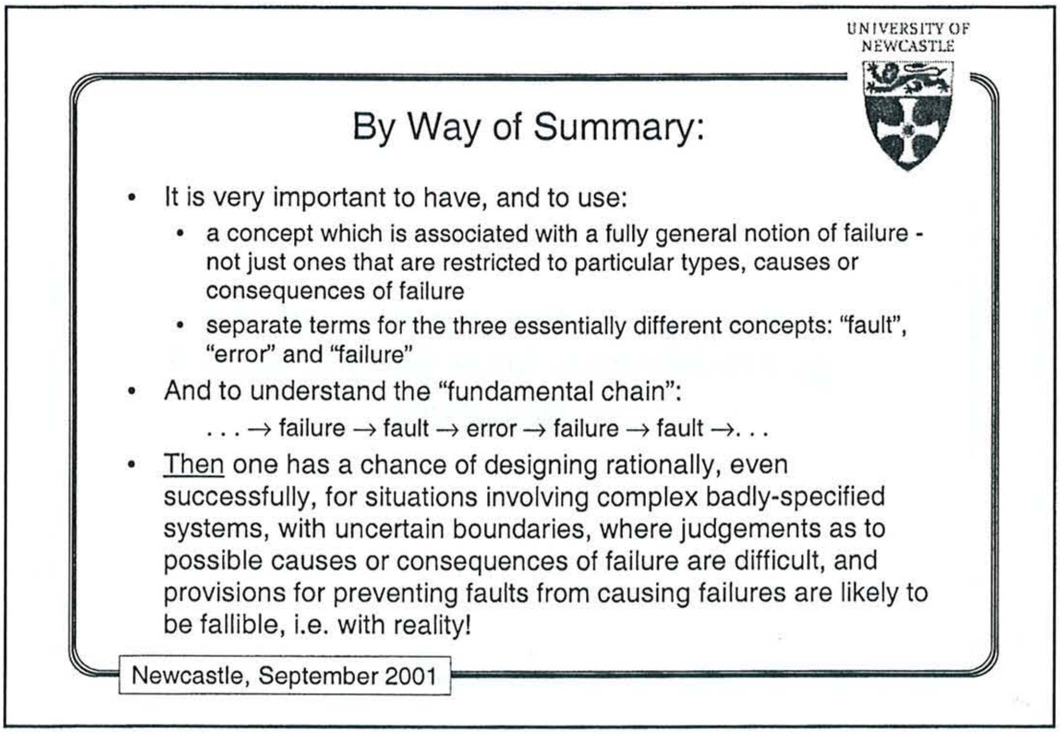


UNIVERSITY OF NEWCASTLE

And now *deja vu* - thirty years on

- The EU IST project MAFTIA - “Malicious- and Accidental- Fault-tolerant Internet Applications”:
  - Concerns systems that should ideally remain operational, protecting all confidential information from unauthorised access, in spite of malicious faults, i.e., attacks, as well as accidental faults.
  - The main objective of MAFTIA is thus to investigate the tolerance paradigm in security.
  - It is assumed that attacks can happen, and some of them can be locally successful. But the overall system should nevertheless remain dependable, even if some subsystems are successfully attacked.
  - Partners: University of Newcastle upon Tyne - UK; FCUL, Lisboa - Portugal; DERA, Malvern - UK; Universität des Saarlandes - Germany; CNRS-LAAS - France; IBM Zurich Research Lab - Switzerland

Newcastle, September 2001



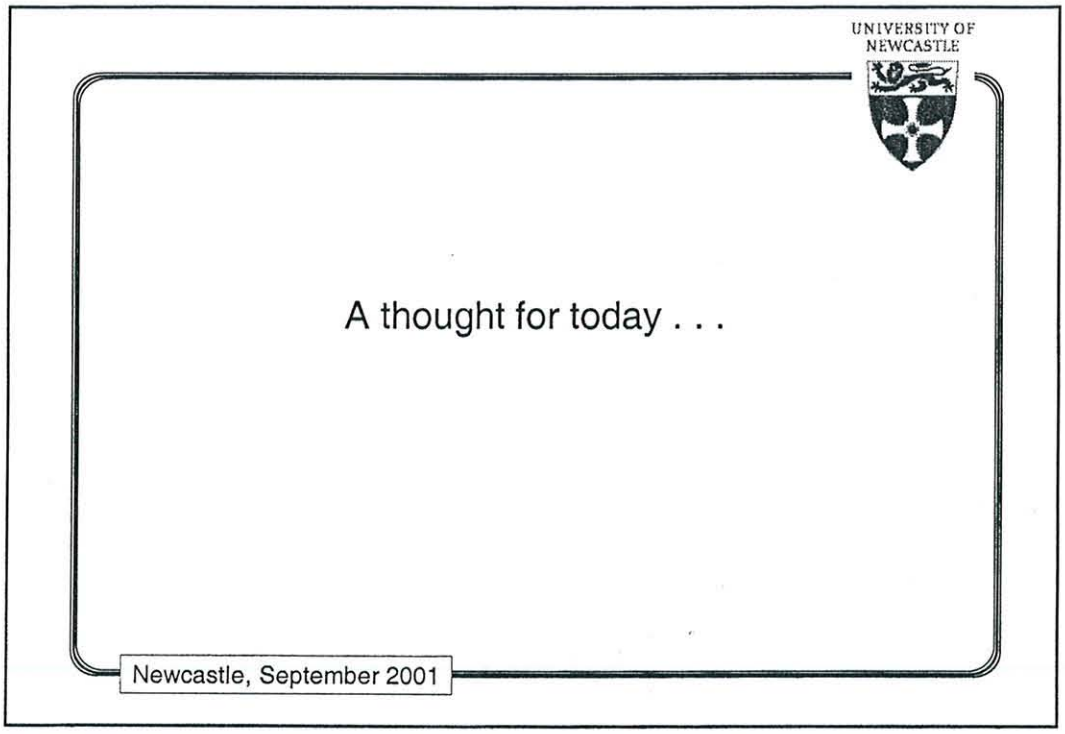
UNIVERSITY OF NEWCASTLE

By Way of Summary:

- It is very important to have, and to use:
  - a concept which is associated with a fully general notion of failure - not just ones that are restricted to particular types, causes or consequences of failure
  - separate terms for the three essentially different concepts: "fault", "error" and "failure"
- And to understand the "fundamental chain":

  . . . → failure → fault → error → failure → fault →. . .
- Then one has a chance of designing rationally, even successfully, for situations involving complex badly-specified systems, with uncertain boundaries, where judgements as to possible causes or consequences of failure are difficult, and provisions for preventing faults from causing failures are likely to be fallible, i.e. with reality!

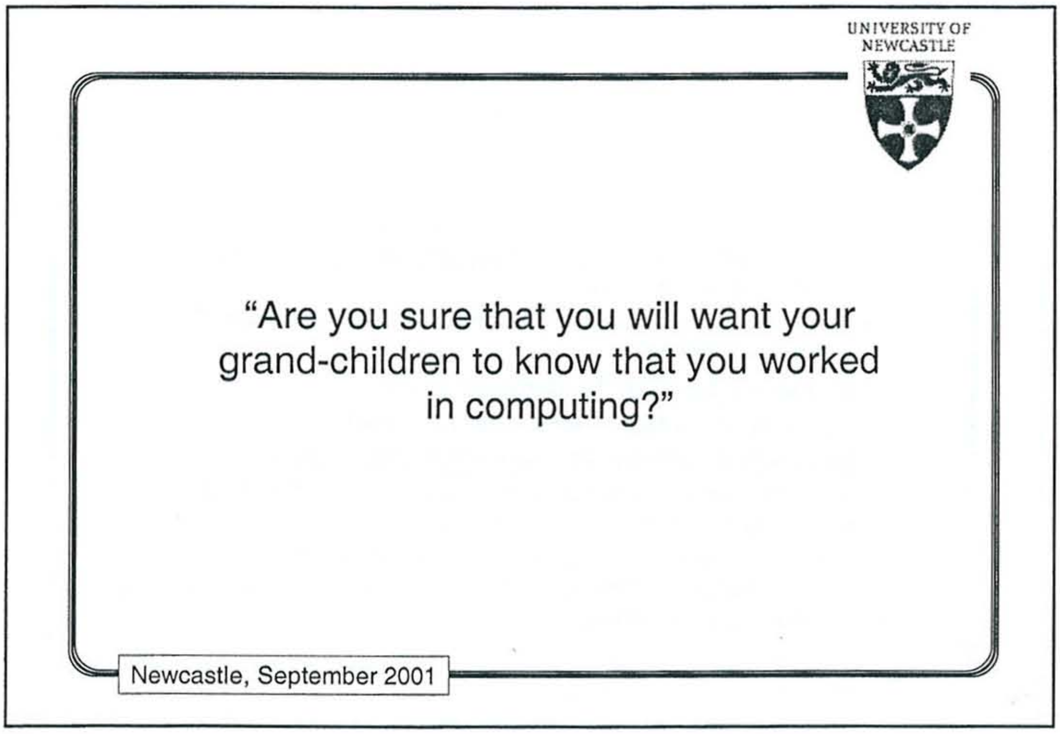
Newcastle, September 2001



UNIVERSITY OF NEWCASTLE

A thought for today . . .

Newcastle, September 2001



UNIVERSITY OF NEWCASTLE

"Are you sure that you will want your grand-children to know that you worked in computing?"

Newcastle, September 2001

DISCUSSION

Rapporteur: Professor M Koutny

Mr Warne asked whether Professor Randell's research was focused on the problem of malicious attacks, which can change a system so that it 'thinks' that it works without errors, even though it is in fact not working correctly.

Professor Randell answered that his own research did not address this issue directly, but that there are approaches in which a solution is derived by attempting to define a very small central core of a system that cannot be corrupted. In any case, he stressed that one has to make some fault assumptions before a solution can be found. Another aspect of this problem, pursued within the MAFTIA project, is the design of dependable systems where different subsystems do not trust that other subsystems work correctly, and therefore take appropriate measures.

Professor Malek stated that, at the present time, the general public is not aware of the issues and problems relating to dependability. He contrasted this with the situation in the area of computer performance, where a new advancement in processor speed can find its way to the headlines of national newspapers. He then asked what could be done in the future to improve this situation.

Professor Randell answered that the best way seems to be to 'frighten' the general public. He then recalled the case of the Y2K problem, which mobilised a huge amount of effort and led to a very successful preventative measures. This success has in turn led to voices that the cost of the whole operation was excessive. Thus, in some sense 'success breeds failure', and 'failure breeds success'. But, in general, many systems have been improved because there were failures in the first place.

Professor Kopetz asked what, in Professor Randell's view, was the impact of academic based research projects on the industrial practice.

Professor Randell answered by giving an example of software development processes within Microsoft, which have been carried out with the help of several fault tolerant techniques developed in academia, such as error recovery schemes and fault masking. His point was that such techniques make significant though still insufficient impact, but at the same time this fact is not in widely publicized.