Software Reliability I

Content

- Basic concept of software reliability
- Various approaches towards software reliability
- Fault tolerance techniques
  - N-version programming
  - Data diversity

Reliability

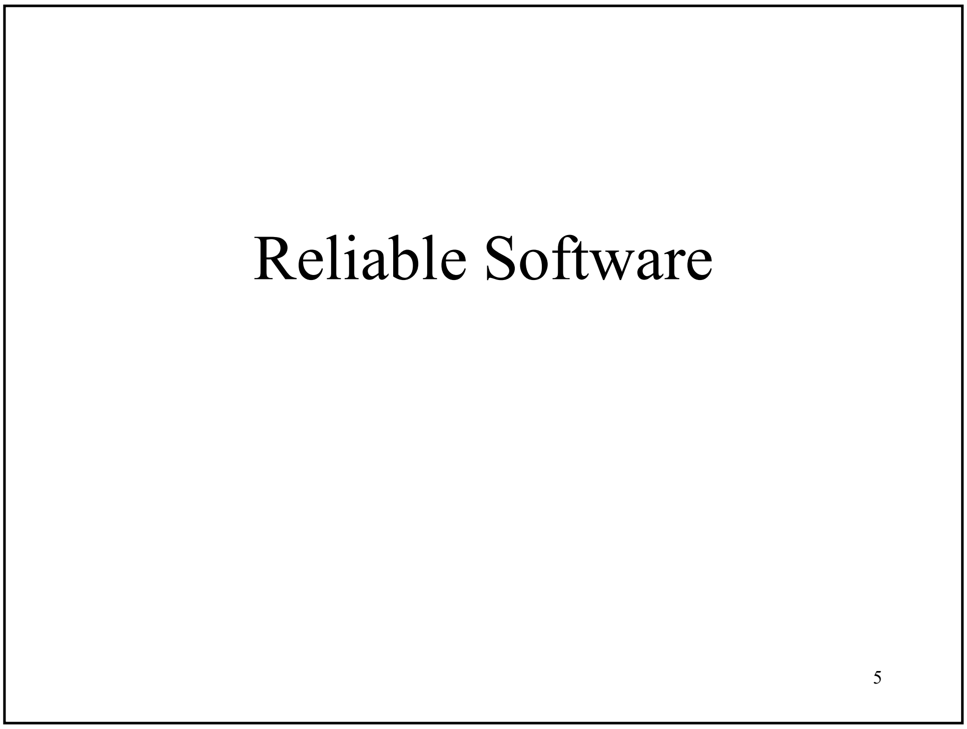
An informal perspective

- reliable hardware
- reliable software

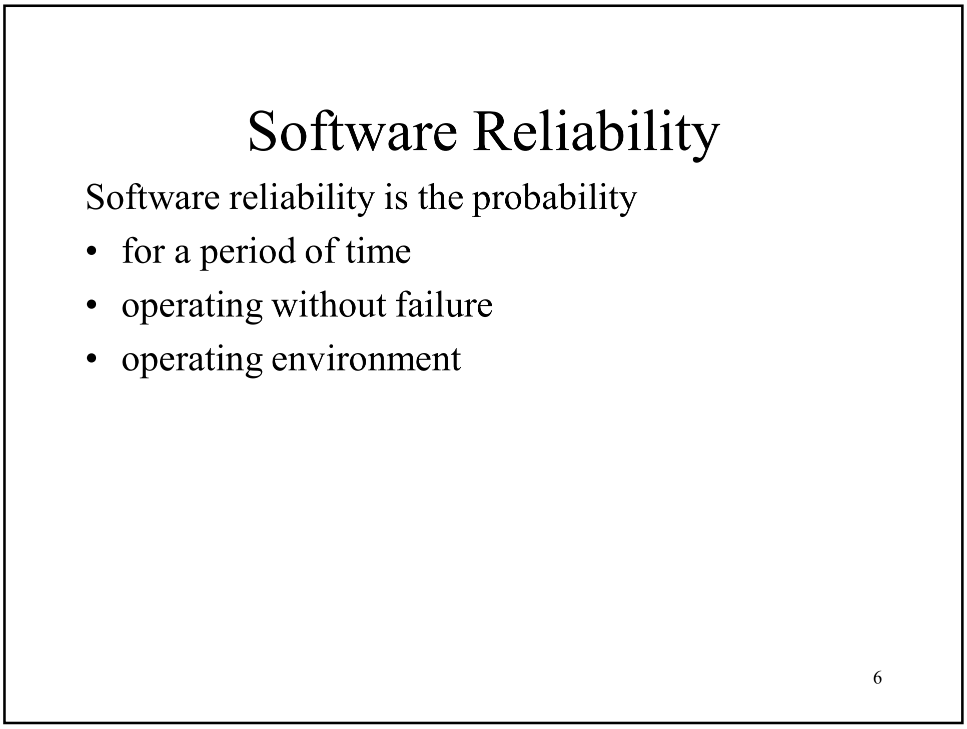
Reliability

Differences in the nature and characteristics between hardware and software

- physical versus logic
- production cost
- design cost



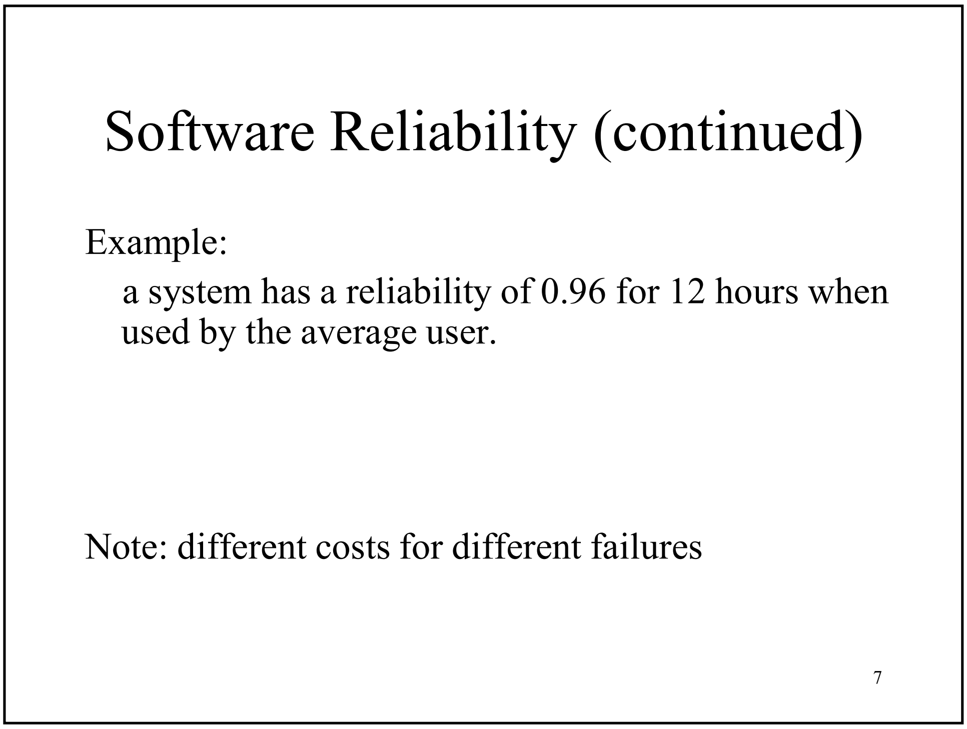
Reliable Software



Software Reliability

Software reliability is the probability

- for a period of time
- operating without failure
- operating environment



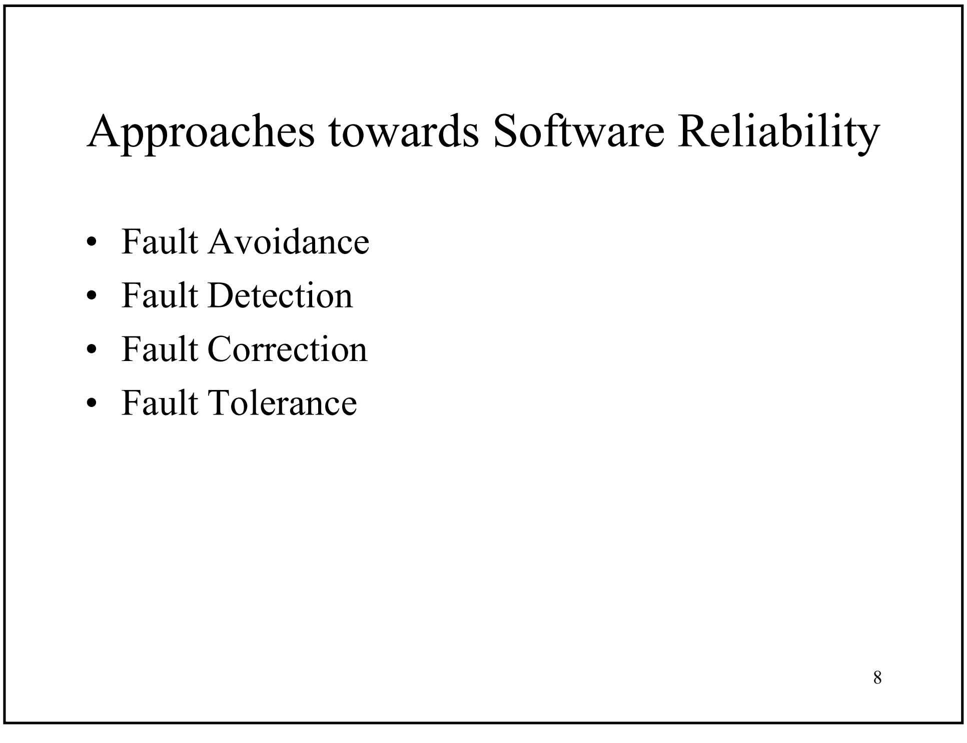
Software Reliability (continued)

Example:

a system has a reliability of 0.96 for 12 hours when used by the average user.

Note: different costs for different failures

7



Approaches towards Software Reliability

- Fault Avoidance
- Fault Detection
- Fault Correction
- Fault Tolerance

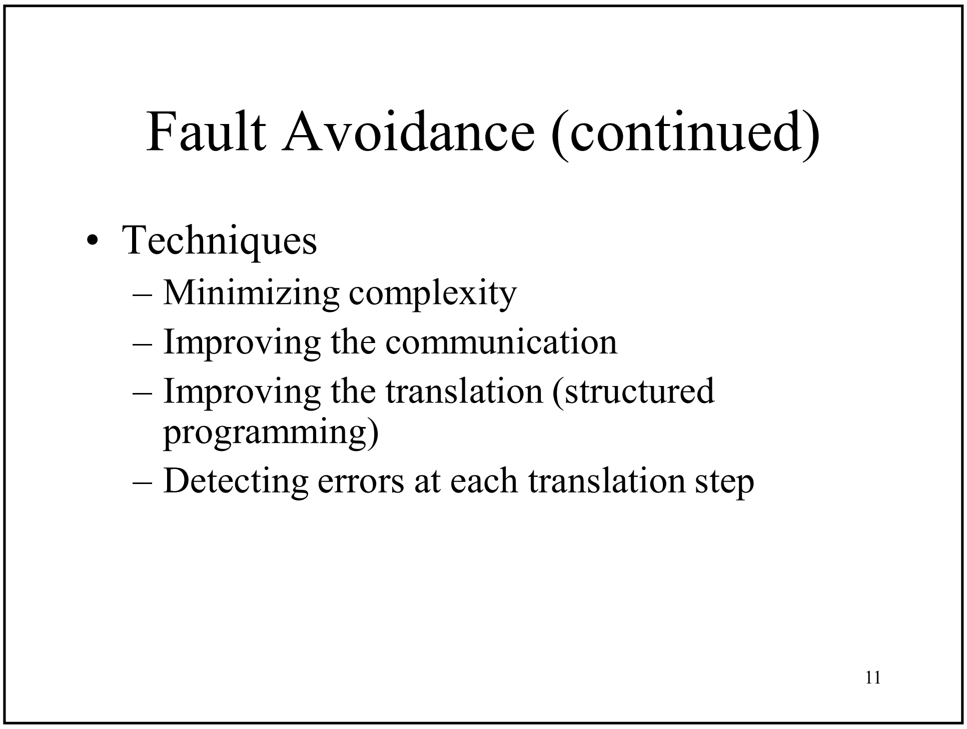
8

Approaches towards Software Reliability (continued)

- Fault Avoidance
  - Minimizing faults
- Fault Detection
  - Revealing faults
- Fault Correction
  - Correcting faults or their damage
- Fault Tolerance
  - Ability to continue operation in the presence of faults

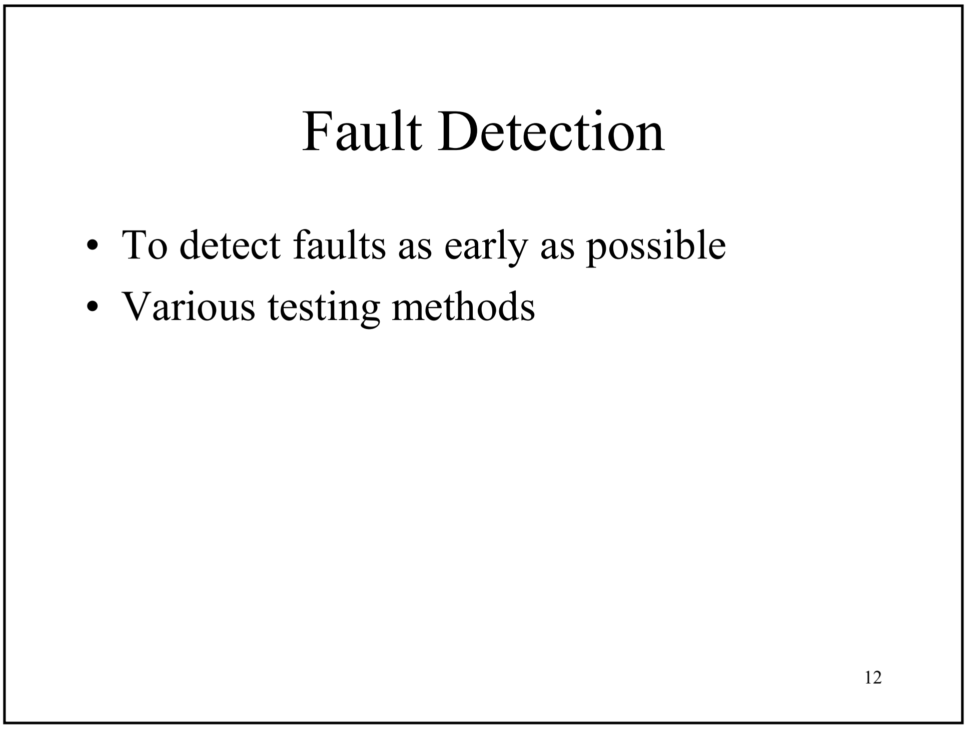
Fault Avoidance

- The optimal approach towards software reliability
- Most well developed



Fault Avoidance (continued)

- Techniques
  - Minimizing complexity
  - Improving the communication
  - Improving the translation (structured programming)
  - Detecting errors at each translation step



Fault Detection

- To detect faults as early as possible
- Various testing methods

Fault Correction

- Fault localization methods
- Program repairing methods
- Less well developed as compared with fault avoidance and fault detection

Fault Tolerance

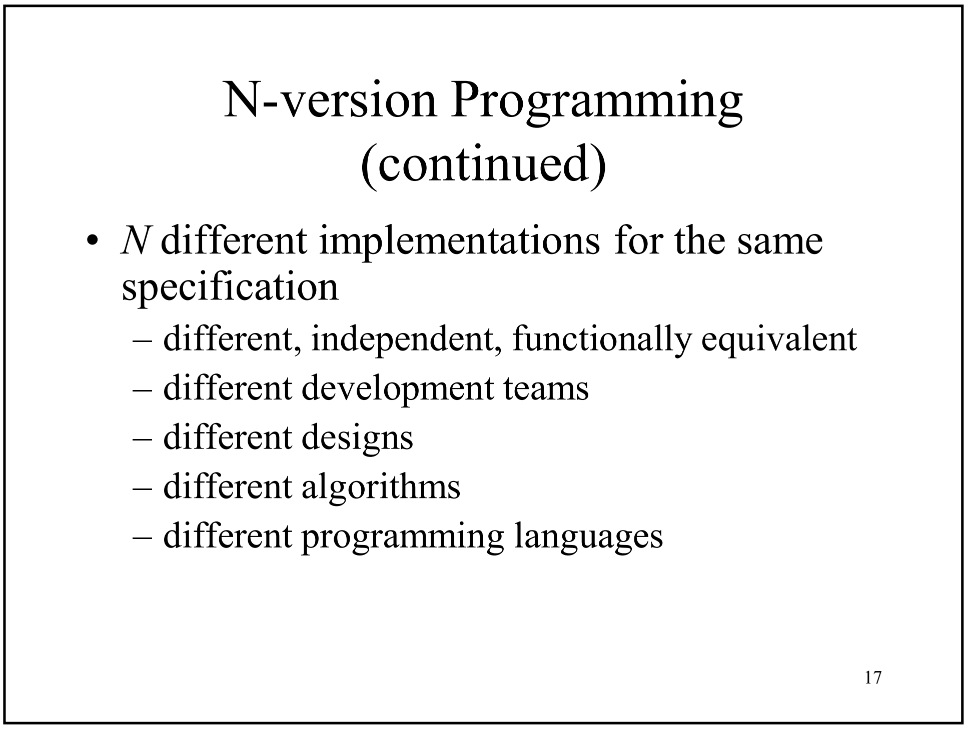
- Error Isolation
  - Isolate problematic functions
- Fallback
  - Commonly used in operating systems
- Redundancy
  - Concept of duplications (from hardware perspective)

Fault Tolerance (continued)

- Redundancy
  - N-version programming
  - Data diversity

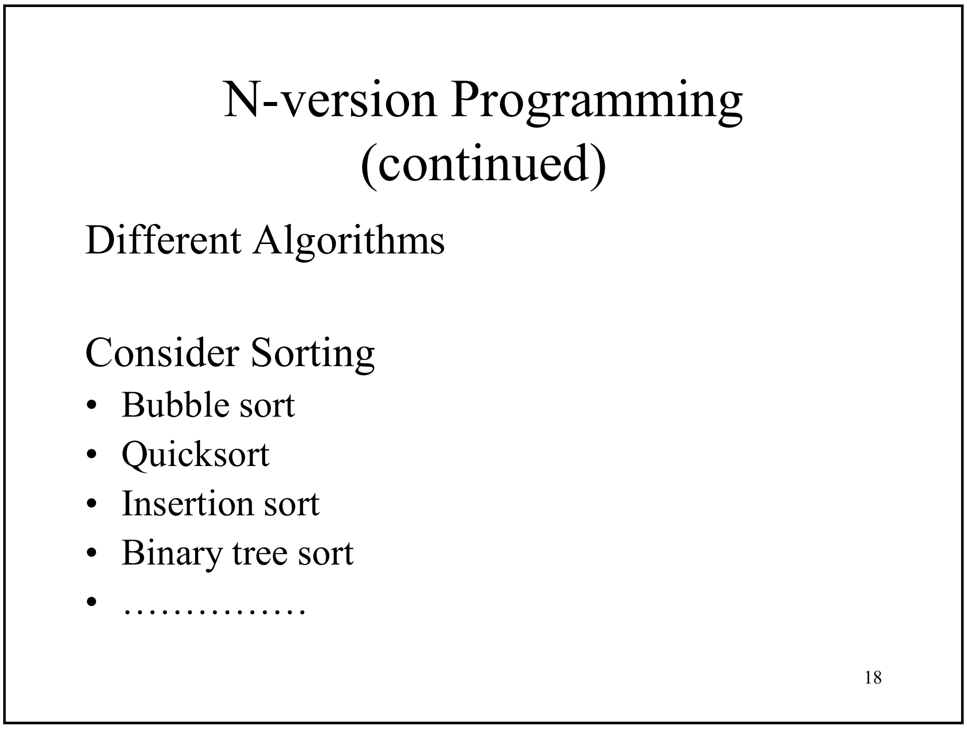
N-version Programming

- Multiversion programming
- *N* different implementations for the same specification
  - differences between hardware and software
  - design diversity
  - assumption of different mistakes



N-version Programming (continued)

- *N* different implementations for the same specification
  - different, independent, functionally equivalent
  - different development teams
  - different designs
  - different algorithms
  - different programming languages

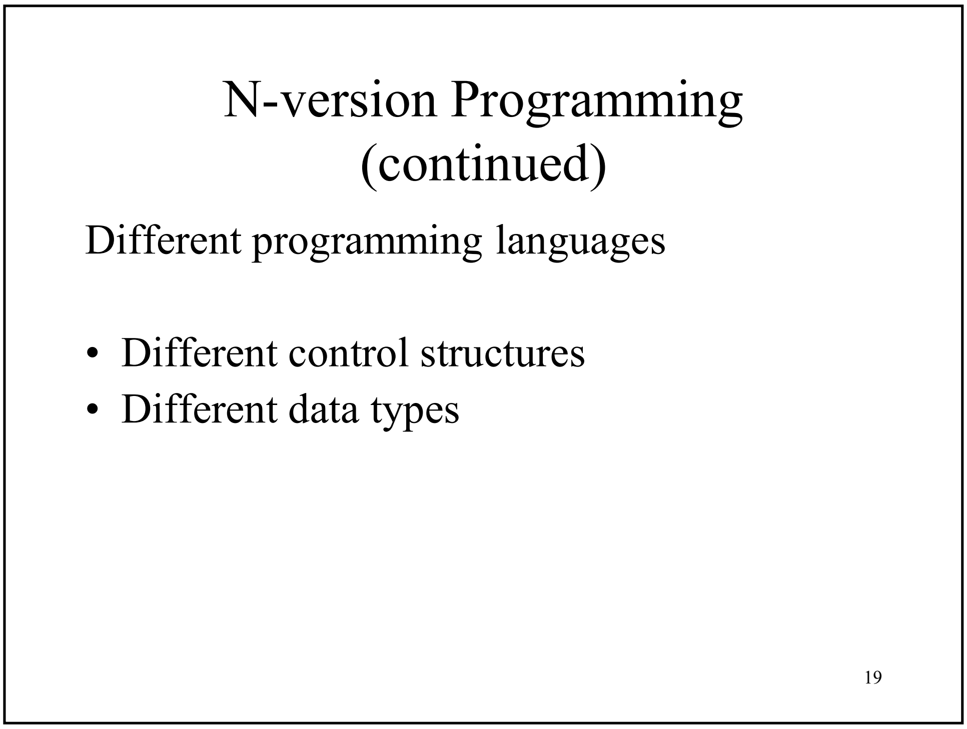


N-version Programming (continued)

Different Algorithms

Consider Sorting

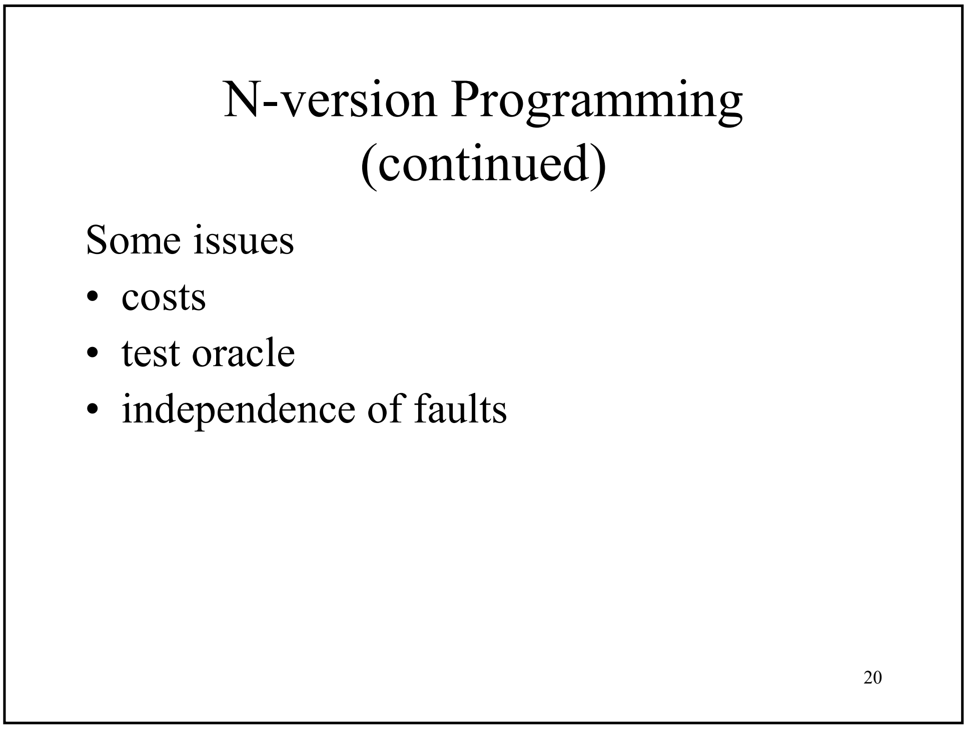
- Bubble sort
- Quicksort
- Insertion sort
- Binary tree sort
- ……………



N-version Programming (continued)

Different programming languages

- Different control structures
- Different data types



N-version Programming (continued)

Some issues

- costs
- test oracle
- independence of faults

Data Diversity

- One implementation
- Data reexpression

Data Diversity (continued)

Suppose in executing the sin program with input 1.3, we have

sin(1.3) = 2.5

Data Diversity (continued)

Consider $\sin(x)$

Re-express $x = a + b$

$$\begin{aligned} \sin(x) &= \sin(a + b) \\ &= \sin(a)\sin(\pi/2 - b) + \sin(\pi/2 - a)\sin(b) \end{aligned}$$

Data Diversity (continued)

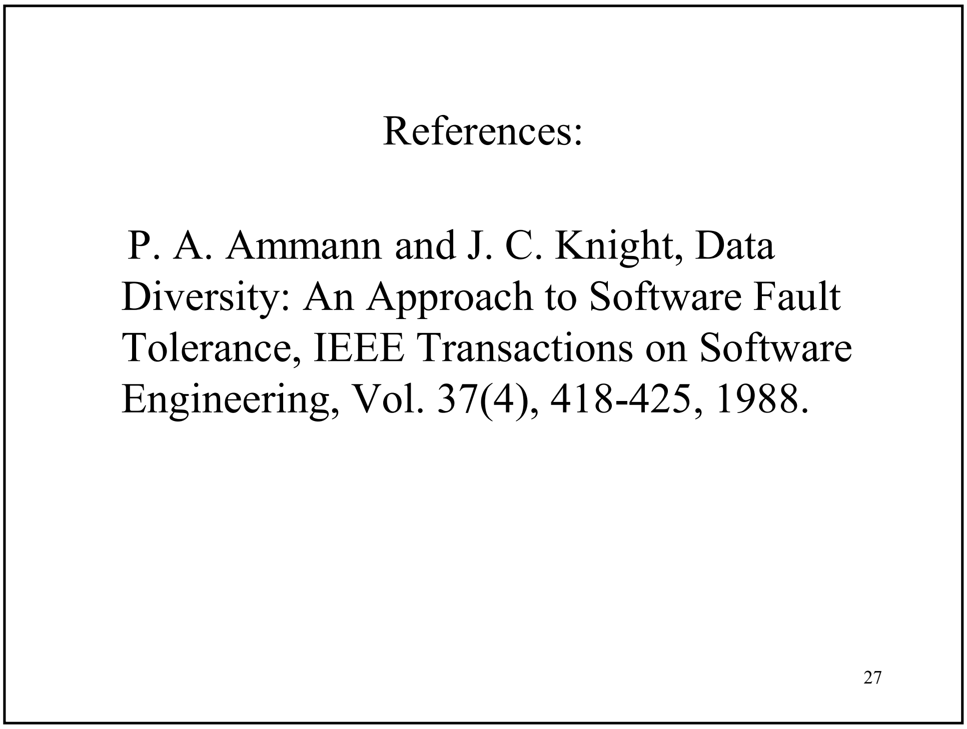
Re-express $1.3 = 0.9 + 0.4$

$$\begin{aligned} &\sin(1.3) \\ &= \sin(0.9 + 0.4) \\ &= \sin(0.9)\sin(\pi/2 - 0.4) + \sin(\pi/2 - 0.9)\sin(0.4) \\ &= \sin(0.9)\sin(1.5708 - 0.4) + \sin(1.5708 -0.9)\sin(0.4) \\ &= \sin(0.9)\sin(1.1708) + \sin(0.6708)\sin(0.4) \end{aligned}$$

Relationship between Metamorphic Testing and Data Diversity

Data Diversity is a special case of Metamorphic Testing

Summary



References:

P. A. Ammann and J. C. Knight, Data Diversity: An Approach to Software Fault Tolerance, IEEE Transactions on Software Engineering, Vol. 37(4), 418-425, 1988.