

# A Vision for the Science of Computing.

Tony Hoare.

BCS International Academic Conference

23 Sept 2008



# The Vision

Computer software contains no more errors



# The Vision

- Computer software contains no more errors
  - it is the most reliable component of any device that contains it



# The Vision

Computer software contains no more errors

- software is the most reliable component of any device that contains it

Programmers make no more mistakes



# The Vision

Computer software contains no more errors

- software is the most reliable component of any device that contains it

Programmers make no more mistakes

- programs work the first time that they run
- and forever after, even when changed

▪



# The Vision

- Programming is an Engineering discipline



# The Vision

- Programming is an Engineering discipline
  - respected for its delivered benefits
  - and for its foundation on basic science



# The Vision

- Software is a branch of Engineering
  - respected for its delivered benefits
  - and its foundation on basic science
- Computing is a branch of pure science



# The Vision

- Software is a branch of Engineering
  - respected for its delivered benefits
  - and for its foundation on basic science
- The basic science of computing
  - has the same ideals and goals, as all other branches of pure science
  - and uses the same methods
  - theory supported by experiment.



# The Insight

- Computer programs are mathematical formulae
  - they do not suffer from rust, fatigue, wear, decay, pollution,...
- Their correctness is a mathematical conjecture
  - to be proved by logic and calculation
  - checked by the computer itself



# Acknowledgements

- Aristotle
- Euclid
- Leibnitz
- Frege
- Russell
- von Neumann
- Turing



# The Science of Computing

is a branch of basic science



# Basic Science

- answers fundamental questions
- pursues scientific ideals
- formalises unifying theories
  - expressed in the language of mathematics
- accumulates convincing evidence
  - by repeatable scientific experiment
- develops powerful tools and apparatus
  - for use by other scientists and engineers



# Basic Science

is driven by curiosity

to answer simple questions:

What does it do?

How does it work?

Why does it work?

How do we know?

**whatever** it **is** – a plant, an animal, a car,...



# Questions about programs

- What does it do?
  - answered by its behavioural specification
- How does it work?
  - answered by its internal interface contracts
- Why does it work?
  - answered by programming theory
- How do we know?
  - confirmed by logical/mathematical proof



# Ideals in Basic Science

Physics: accuracy of measurement

Chemistry: purity of materials

pursued for the benefit of other scientists

for the sake of scientific glory

far in advance of commercial need



# Ideals in Basic Science

Physics: accuracy of measurement

Chemistry: purity of materials

**Computing Science: zero defect programs**

pursued for the benefit of other scientists

for the sake of scientific glory

far in advance of commercial need

# Ideals in the professions

- health for the doctor
- justice for the lawyer
- correctness for the programmer
- professional ideals contribute to Society's confidence in the integrity of the profession.



# Unifying theory

- basic science seeks unifying theories
  - the four fundamental forces of physics
- explaining diverse phenomena
  - gravitation in planets and apples on trees
- supported by evidence
  - accumulated from all the unified theories



# Unifying Theory of programming

- formalises our understanding of
  - objects, pointers, exceptions,
  - classes, methods, interfaces, inheritance,
  - concurrency, weak memory models,
  - distribution, communication,
  - configuration and reconfiguration
  - transactions, queries, spreadsheets, ...



# Unifying Theory of programming

- extends to many languages
  - functional and procedural,
  - data base and query,
  - spreadsheets and scripting,
  - special-purpose, application-dependent
  - specification and design



# Diversity of application

- theory tested by experiments on
  - web services, desktop applications,
  - compilers, linkers, program generators,
  - common base libraries,
  - OS kernels, drivers, file systems,
  - embedded control systems,
  - smart cards, remotes, cell-phones,...



# Experimental Verification

- will establish relevant properties
  - no overflow, no exceptions, no memory leak
  - no deadlock, no race conditions, no looping
  - no violation of interface contracts/protocols
  - in the ideal: total functional correctness
- on varying scales
  - laboratory models, realistic programs
  - parts of real systems



# Software Engineering Toolsets

- based on unified theory
- covering all aspects of program lifecycle
  - domain models, requirements, specifications,
  - program generation, patterns, re-use,
  - architectures, designs, interfaces,
  - coding, inspections, testing, delivery,
  - maintenance, modification, evolution,
  - de-commissioning,...



# Big Science

- Projects in big science
  - last for decades,
  - involve thousands of scientists,
  - theorists, tool-builders, experimenters
  - pursuing a coherent vision
  - for the advancement of science.



# Big Science

- Physicists build colliders
- Astronomers build telescopes
- Biologists decode the human genome.
- Computer Scientists?
  - will we rise to the challenge?



if we do...

Software will contain no more errors  
– than any other engineering product

Programmers will make no more mistakes  
– than any other professional engineer



# The Hope (2020 – 2050?)

Software will contain no more errors  
– than any other engineering product

Programmers will make no more mistakes  
– than any other professional engineer

Cost of program error will be reduced  
– saving \$10 billion per year, worldwide



# \$100 billion (approx)

the world-wide annual cost of software error.

60% falls on developers, 40% on users.

Estimate based on survey of US industry

Planning report 02-03, prepared by NIST for

US Department of Commerce, May 2002

‘The economic impacts of inadequate  
infrastructure for software testing.’