

The Ideal of Verified Software

Tony Hoare

FM05 Newcastle

July 18, 2005

A mature scientific discipline

- Sets its own agenda
 - Stimulates and satisfies its own curiosity
 - Poses its own fundamental questions
- ... what about the Science of Computing and the Engineering of Software?

In Engineering we ask...

- What does the product do?
 - the specification tells us
- How does it work?
 - its internal interface specifications tell us.

In Science we ask more...

- Why does it work?
 - The underlying scientific theory tells us.
- How do we know the answers are right?
 - By calculation and by convincing experiment.

A Program Verifier

- confirms validity of an explanation why a program works,
- by checking consistency of code with internal and external specifications.
- It is a basic experimental tool for research into the science of programming.
- Its construction is still a grand challenge to Computer Science

The Human Genome project (1991-2004)

- had a clear set of deliverables,
- a planned route to application,
- and potential for great benefit.
- It pursued scientific ideals
- free from commercial pressures.
- It built on the current state of the art
- It won public appeal.
- It changed the course of Science.

The Verified Software project

- is modelled on the Human Genome project
- and shares many of its properties

Clear deliverables

- A comprehensive theory of programming
 - concurrency, object orientation,...
- A coherent toolset based on the theory
 - development aids, verifiers, test case generators,...
- A wide-ranging collection of mechanically verified programs
 - from safety-critical and embedded codes to open source libraries, middleware and desktop applications
 - verified automatically to high levels of safety, security, soundness, serviceability, functional correctness.

Route to application

- Verified programs will replace existing versions in daily use
 - so subsequent evolution maintains correctness.
- Verification technology will be integrated into commercial toolsets
- The cost associated with program error will be significantly reduced

Potential benefit

- Reduction in avoidable program errors could even now save \$22 to \$60 billion per year in US.
- “The Economic Impacts of Inadequate Infrastructure for Software Testing”
 - (US Dept. Commerce Planning Report 02-03, May 2002).

Scientific ideals

- Academic research pursues ideals
 - purity of materials,
 - accuracy of measurement,
 - generality of theory,
 - and correctness of programs
- far beyond the current needs of the market place

Commercial development

- of program analysis/design tools
 - appeal to current educational levels
 - and so must concentrate on the pictures.
- Commerce must follow market demand
 - to discover more faults in existing programs
 - until they are too dangerous to remove.
- Further progress will need verification
 - to avoid errors rather than detect them

State of the art

- Smart-card applications have been manually proved (eg. Logica).
- Safety-critical systems have been developed from specification (eg. Praxis).
- Commodity software already includes many assertions (eg. Microsoft Office)
- Open Source software is freely available for research, as well as for use (eg. Apache).
- Programming theory covers O-O, concurrency (eg. Separation Logic, Process algebra ,...)

Some Available Tools

- Program analysers and optimisers
- Abstract Syntax Tree compilers
- Verification Condition Generators
- Program Development Environments
- Assertion generators Abstract interpreters
- Proof searchers Constraint solvers
- Decision procedures Model checkers
- Algebraic simplifiers

The Challenge needs

- Large and long-term commitment
 - to collaboration as well as competition
 - on an international scale.
- New links between established research schools, conference series, journals...
- New criteria for refereeing, research grant evaluation, personal promotion,...
- To change the course of Science
we need a 'Grand Challenge'

Public appeal

- Reduction in software failures and errors
 - will win over-due respect for the profession
- The idea of self-checking computers
 - will excite the curiosity of students
 - and attract interest from researchers in logic, mathematics and philosophy as well as computing

IFIP Working Conference

- Verified Software:
Theories, Tools and Experiments
- Zurich, October 10-14, 2005
- Chairmen: Jay Misra and Tony Hoare
- Organisers: Bertrand Meyer, Jim Woodcock, Natarajan Shankar

A Program Verifier

One can dream of routinely using a verifying compiler as an everyday tool. In the context of this idea our work has been extremely modest and must be considered as a small first step. We only hope that, indeed, this has been a first step of a progression which will allow this dream to come to fruition.

A Program Verifier
Thesis by James C. King
Carnegie Institute of Technology
September 1969