1,000,000 Lines of Verified Code

Jim Woodcock
University of York
September 2006

A grand challenge

- Tony Hoare

  > **automatically verified software:**
  > **a grand scientific challenge for computing**

- UK EPSRC-funded meetings, US NSF-funded meetings
- UK-China network proposal
- Zürich conference `vstte.inf.ethz.ch`
- Macau conference
- FACJ article, 2006
- IEEE Computer articles, April 2006, October 2006
- research roadmap `qpq.csl.sri.com`
- European dimension?

Hoare's Verification Grand Challenge

A mature scientific discipline should set its own agenda and pursue ideals of purity, generality, and accuracy far beyond current needs

what should we do?

- *achieve a significant body of verified programs*
- *precise external specifications*
- *complete internal specifications*
- *machine-checked proofs of correctness*

Deliverables

a collection of verified programs

- 1,000,000 lines
- replacing existing unverified ones
- continuing to evolve as verified code
- **a repository**

You can't say any more it can't be done!
Here, we've done it!

First step: research roadmap

roadmap should set out long-term co-ordinated programme of incremental research

1. **pilot projects** to evaluate feasibility and guide technology development
2. **large-scale experiments** that benchmark the technology

Goals of the Repository

1. accelerate development of verification technology
2. provide focus for verification community
3. provide open access
4. collect challenging applications
5. identify key metrics
6. enumerate challenge problems
7. standardise formats
8. define quality standards

A pilot project: Mondex

- *year-long pilot project launched in January 2006*
- *demonstrate research collaboration and competition*
- *generate artefacts to populate the Repository*
- ***verify key property of Mondex smart card***
  - *financial security*
- *assess current state of proof mechanisation*

Mondex

- *electronic purse hosted on a smart card*
- ***developed to high-assurance standard ITSEC Level E6***
- *consortium led by NatWest, a UK high-street bank*
- *purses interact using communications device*
- *strong guarantees needed that transactions are secure*
- *in spite of power failures and mischievous attacks*
- *electronic cash can't be counterfeited*
- *transactions completely distributed: no centralised control*
- *all security measures locally implemented*
- *no real-time external audit logging or monitoring*

The original verification

- *seriously security critical*
- *Logica (and Oxford) used Z for development process*
- *formal models of system and abstract security policy*
- *hand proofs that system design possesses security properties*
- ***abstract security policy specification about 20 pages of Z***
- ***concrete specification (n-step protocol) about 60 pages***
- *verification suitable for external evaluation*
  - ***about 200 pages of refinement proof***
  - ***100 pages of derivation of refinement rules***

The original proof

- *carefully structured for understanding*
- *much appreciated by Mondex case study groups*
- *original proof vital in successfully getting required certification*
- *also useful in finding and evaluating different models*
- *original team made key modelling discovery*
- *abstraction gave precise security property*
- *explained why protocol is secure*

The original proof

- revealed a bug in implementation of secondary protocol
- failed proof explained what had gone wrong
- convincing counterexample that the protocol was flawed
- insight to change design to correct it
- third-party evaluators also found a bug:
  - an undischarged assumption

The challenge

- *sanitised version of Mondex documentation publicly available*
  - *Z specifications of security properties*
  - *abstract specification*
  - *intermediate-level design*
  - *concrete design*
  - *rigorous correctness proofs of security and conformance*
- *originally no question of mechanising proofs:*

  "mechanising such a large proof cost-effectively is beyond the state of the art"

- *challenge: investigate the degree of automation that can now be achieved in the correctness proofs*

The players

- Alloy (MIT)
- Event-B (Southampton)
- OCL (Bremen)
- PerfectDeveloper (Escher)
- Raise (Macao/DTU)
- Z (York)
- agreed to work for one year, without funding
- ...separately and silently:
  - a group in Augsburg began work using KIV and ASMs

Two distinct approaches

- **Archaeologists**
  - make as few changes as possible to original documentation
  - shouldn't change models just to make verification easier
  - how would we know that our results had anything to do with the original specification?
- **Technologists**
  - use best proof technology now available
  - these new tools don't work for Z
  - two choices
    - translate existing models into new languages
    - create new models better suited to new tools

Z (York)

- *Leonardo Freitas and Jim Woodcock*
- *Z/Eves theorem prover*
- *mechanise all proofs, remaining faithful to original formalisation*
- *made two changes to make finiteness explicit*
- *progress: succeeded in mechanising most of the project*
- ***taken just over a month to complete***
- *about nine working days using Z/Eves*

Results

- *informal proofs were useful*
- *structure and detail*
- *hand proofs particularly thorough*
- *about 140 verification conditions (VCs) of different complexity*
- *average five proof steps per VC*
- *built-in automation: 200 steps require little interaction*
- *other parts abstracted into general lemmas with some effort*
- *400 intermediate steps require internal knowledge of Z/Eves*
- *100 creative steps require domain knowledge (witnesses)*
- *general theories needed about language constructs*

Results

- *missing properties in intermediate design*
  - *operations involving non-authentic purses are permitted*
- *preliminary findings are very encouraging*
- *Z/Eves theorem prover hasn't changed in ten years*
- *mechanisation could have been carried out during original project*
- *a few weeks of effort required*
- *motivation and expertise lacking, not proof technology*

Raise (Macao/DTU)

- *Chris George and Anne Haxthausen*
- *RAISE method and RSL specification language*
- *high-level abstract specifications*
- *low-level designs, including explicit imperative programming constructs*
- *RSL specifications verified using PVS*

Approach

- *initial RSL specifications transliterations of Z*
- *group felt inhibited:*
- *new models in RSL*
- *abstractly, Mondex is simply as a problem in accounting*
- *no purses, no protocol messages*
- *just three bottom-line values and transfer money operations*
- *middle level: abstract purses and concrete operations*
- *no details of mechanisms preserving asserted invariant*
- *each operation proved correct wrt abstract specification*
- *concrete level: full details of value-transferring protocol*
- *each operation proved to implement its middle version*

Results

- current specification is tenth version
- 2,200 lines of RSL in 13 files, with 366 proofs
- 180 proofs fully automatic
- 300 prover commands for typical concrete invariant proof
- 150 commands to prove concrete invariant implies abstract
- difficulties proving finiteness
- large amount of reworking of models
- didn’t benefit from using original modelling details
- biggest problem experienced was finding suitable invariant
- subtle trade-off between refinement and invariant proofs
- RSL group turned out to be technologists

Escher (PerfectDeveloper)

- David Crocker
- tool for rigorous development of computer programs
- correctness-by-construction paradigm
- component interfaces are verified by static analysis
- certain that the components will strictly conform to their contracts at run-time
- object-oriented style, producing code in both Java and C++
- objective: fully automatic proof and implementation in Java
- learn more about system-level specifications in Perfect
- understand and overcome limitations of PD prover

Approach

- *technologist, but specifications are recognisable translations*
- *PD prover fully automatic, so details of proofs are hidden*
- *don't obviously follow originals*
- *refinement steps had to be revised to be suitable for PD*
- *additional assertions provided as hints*
- *where necessary prover was enhanced*
- *working code generated for purse and other components*
- *dropped atomic abstraction of protocol*
- *transactions are fundamentally non-atomic*
- *reformulated security properties*

Results

- 213 VCs, 191 proved automatically
- about 550 lines of Perfect
- proof run takes about six hours
- all successful VCs discharged in less than six minutes each
- team spent about 60 hours on the project

Augsburg (KIV)

- *Gerhard Schellhorn*
- *claims prize of being first to mechanise entire Mondex proof*
- *KIV specification and verification system*
- *demonstrated small errors in rigorous hand-made proofs*
- *alternative formalisation of communication protocol in ASM*
- *technologists, but with some archaeology*
- *models and proofs are clearly inspired by original work*
- *mechanical verification of full Mondex case study*
- *except transcription of failure logs to central archive*
- *orthogonal to money-transfer protocol*

Results

- mimicked data refinement proofs faithfully
- work completed in four weeks
- one week to get familiar with the case study and specify ASMs
- one week to verify proof obligations of correctness and invariance
- one week to specify the Mondex refinement theory
- one week to prove data refinement and polish for publication
- existence of (nearly) correct refinement relation helped
- time needed to find invariants and refinement relations
- shorter in ASM!

Results

- *main data refinement proofs require 1,839 proof steps with 372 interactions*
- *interesting, both technically and organisationally*
- *group worked independently*

Next Steps

- *Mondex case study shows that verification community is willing to undertake competitive and collaborative projects*
- *...and that there is some value in doing this*
- *collection of papers in special FACJ issue*
- *detailed comparison of results*
- *curation of key parts of experiments*

join the next project!

A challenge for the European Academy

- *Joshi & Holzmann's space-flight file store*
- *verified implementation on flash memory*
- *POSIX interface*
- *Morgan & Sufrin's* Unix *filing system in Z*
- *Synergy file store (Z + ACL2)*
- *European project:*
  - *verification from top to toe*
  - *file structure refinement*
  - *implementation in hashmaps*

let's do it!