Journal of Universal Computer Science, vol. 13, no. 5 (2007), 661-668
submitted: 7/5/07, accepted: 25/5/07, appeared: 28/5/07 © J.UCS

The Verification Grand Challenge

Jim Woodcock
(Dept. of Computer Science, University of York,
York YO10 5DD, UK
jim@cs.york.ac.uk)

Richard Banach
(School of Computer Science, University of Manchester,
Manchester M13 9PL, UK
banach@cs.man.ac.uk)

Abstract: This paper overviews the *Verification Grand Challenge*, a large scale multinational intiative designed to significantly increase the interoperability, applicability and uptake of formal development techniques. Results to date are reviewed, and next steps are outlined.
Key Words: Formal Methods, Verification Grand Challenge
Category: D.2.4, F.3.1, B.2

1 Introduction

In today's world, the cost of malfunctioning software is truly staggering. According to a report of the US Department of Commerce's National Institute of Standards and Technology, the cost to the US economy alone could be as high as $60Bn a year. If you multiply by three to include the impact on the world as a whole, you reach a figure that exceeds the gross national product of many countries. If the manufacturers of everyday goods, such as cars, sold their products in the way that sofware vendors sell theirs, i.e. without any waranty that is worth anything to the purchaser, they would quickly go out of business. The situation with software is clearly unsatisfactory.

What can be done about this? Formal techniques, for many years the almost exclusive province of computer science researchers, and in the teeth of many who claimed that they were impractical for industrial sized applications, are reaching a level of usability that will enable software manufacturers to issue meaningful guarantees for their products. It is forseen by many people working in this field, that this will prove to be not only the cheapest way of achieving this level of reliability, but possibly also the only way of doing so routinely. As this vision slowly becomes a reality, the somewhat hit-and-miss character of conventional software engineering, with its dependence on unreliable testing as a means of discovering the flaws created by an unpredictable software construction process, will be replaced by a software construction process with predictable properties, comparable with most mature engineering disciplines.

This short paper overviews a 'Grand Challenge' programme of work, initiated by the Formal Methods community, whose goal is to accelerate the maturation of formal techniques to the point where they are seen by the majority of the software industry, as an ingredient in their activities that not only delivers, but is eminently usable, and is financially convincing.

2 The Grand Challenge

The Verification Grand Challenge (VGC) [QPQ-DSR, VST (2005)], is an ambitious, international, long-term research programme that seeks to create a substantial and useful body of of code that has been verified to the highest standards of rigour and accuracy. It has a 15 year perspective, hoping in that time, to achieve three major objectives:

• to establish a unified framework within which different theories of program construction and verification may co-exist and communicate productively;

• to build an integrated suite of tools to support all aspects of verified software construction: requirements capture, specification, validation test-case generation, refinement, analysis, verification, run-time chacking;

• to populate a repository of formally specified and verified codes, that can not only serve as exemplars to convince others of the utility and practicability of formal techniques, but that also are seen as being useful in practice, and, ideally, that are taken up and used in anger.

The verification of which we speak in principle embraces any properties that are seen as significant for software. There are functional correctness issues such as: conformance to functional specification, type correctness, data consistency, numerical accuracy, absence of runtime errors, termination, translation validation, memory leakage, serialisability etc. There are also many nonfunctional issues such as: dependability, safety, security, timeliness of response, deployability, maintainability etc.

Initially, the VGC can set up the repository and populate it with interesting and useful software artifacts. This will provide a focus round which the remaining objectives can congregate. Thus the existence of a growing family of verified case studies will help to mobilize an international community of researchers, and encourage the creation of tools that can deal with the repository's contents in a uniform and integrated manner, backed up by suitable theory. So the repository is a key early objective. Researchers have already proposed a range of substantial problems to work on, including a verified Apache Web Server, a reference TCP/IP implementation, and the Linux Kernel.

3 A first Case Study: Mondex

To get the ball rolling, an initial case study was proposed for the repository, to be worked on during 2006: the mechanical verification of the Mondex Electronic Purse. Mondex is a smartcard purse application. That is to say it contains (in digital form) real money, that can be traded in the same way as cash. The idea is to make the electronic cash reflect the properties of real cash as closely as possible. Thus, just as for a £10 note, or a \$10 note, or a Rp10 note, since the sole concern of the note (once it is in the field) is to be unforgeable, so for Mondex cash the overriding property is also non-forgeability. Moreover, just as it is *not* the responsibility of the note to be concerned about whether it is being transacted in an honest way, or in a way in keeping with its current owner's intentions, it is *not* the responsibility of Mondex cash to check on who owns it, or whether some transaction is as the Mondex cash's owner thinks it is.

Mondex was developed by NatWest, a UK high-street bank. Since a genuine commercial enterprise (typically a bank such as NatWest), must underwrite the funds contained in a product like Mondex, in order to not undermine commercial confidence in the bank the application must be as secure as possible. Consequently, state-of-the-art techniques of the time (about ten years ago) were used in the Mondex development. This meant conformance to the UK standard for high-assurance systems (the Information Technology Security standard, ITSEC [ITSEC (1991)]), at its highest level, E6. [1]

ITSEC E6 demands that there is an abstract model and a concrete model of the application, and that there is a proof of correspondence between them. The abstract model should reflect the desired properties of the system in the cleanest way possible, while the concrete model should reflect the actual implementation in a credible way. In the case of Mondex, both models were written in the formal specification language Z [Spivey (1992), ISO-Z (2002)], and the proof of correspondence was a hand-performed refinement proof, showing that the actual Mondex *n*-step concrete protocol was mathematically equivalent to an abstract notion of transaction in which the the funds were either transfered instantaneously, or were 'lost in transit' (the latter possibility covering a range of scenarios in which the environment prevented the completion of the protocol in the normal way). Moreover, the properities of the *n*-step protocol were such that even if funds *were* lost in transit, this fact itself was knowable by the system as a whole, so that in such cases, funds could be reliably restored to their rightful owner, in a way that undermined neither the bank's financial solidity, nor customers' confidence.

Following a rather conservative strategy, the original protocol design was done before the formal part of the development was embarked on. The formal analysis revealed a flaw in a secondary protocol, which was then fixed. The third-party evaluators also found a flaw in one of the proofs, which was also fixed. Following that, and somewhat uniquely for a genuine commercially sensitive development, a sanitized version of the Mondex formal development, authored by the principal actors, was put in the public domain [Stepney et al. (2000)]. This reduced version is about 230p. long. A further 100p. document [Cooper et al. (2002)] gives a precise description of the notion of refinement used in the development. At the time, most people believed that gaining additional assurance, by checking the proofs mechanically, was beyond the state of the art — a view that recent events have shown to be overly pessimistic, as noted below. So the goal of the VGC initial case study was to tackle the Mondex formal development using today's tools.

There are essentially two approaches to such a task. One can take the original documentation, and attempt to verify it 'straight out of the box'; one might call this an 'archeology driven' approach. Alternatively, one could attempt to translate the original into a different notation in order to be able to use tools specific to the other notation; one might call this a 'technology driven' approach. In a sense, any approach which does not deal directly with the original Z formulation contains elements of technology.

A number of groups around the world took up one variant or another of the Mondex challenge, using a number of contemporary formalisms and tools: Alloy [Alloy] at

[1] Nowadays, national standards such as ITSEC have been superseded by a worldwide ISO standard called Common Criteria [Common Criteria (2005)]. ITSEC E6 corresponds to the highest Common Criteria level, EAL7.

the Massachusetts Institute of Technology; Event-B [Abrial (2007)] at the University of Southampton; OCL [OCL (2003)] at the University of Bremen; Perfect Developer [Escher] at Escher Technologies; RAISE [RAISE (1995)] at the University of the United Nations Macao and the the Technical University of Denmark; Z [Spivey (1992), ISO-Z (2002)] at the University of York. Another group, at the University of Augsburg, heard about the Mondex endeavour after it had started, and decided to join in, using ASMs [Börger and Stärk (2003)]. We briefly review some of these efforts now.

3.1 Z and Z/Eves at York

In a project such as the Mondex retrospective, there should be at least one group which takes the original documentation at 100% face value, and attempts verification directly. It is thus fitting that at York, the current affiliation of two of the authors of [Stepney et al. (2000)], such a task was undertaken. Jim Woodcock and Leo Freitas worked directly from the documentation in [Stepney et al. (2000)], and attempted a mechanisation using the Z/Eves theorem prover.

With today's experience of program proving, the Z/Eves exercise fell within predictable boundaries. The close adherence to the original hand-crafted proof meant that the greatest challenge in such an undertaking, the discovery of exactly the right invariants relating the different models, was already overcome. As typically happens during full mechanisation, some relatively small gaps in the proof that the humans had overlooked came to light, and were fixed. Other small details, handled efficiently by humans because of their 'obvious' nature, proved nevertheless surprisingly difficult to convince a machine about. However, as a whole, the underlying proof survived mechanical scrutiny unaltered. This archaeological exercise revealed that the whole job could be done in a matter of weeks.

The most notable thing is that Z/Eves has been in existence and has not changed for 10 years or so. One therefore concludes that this mechanisation could have been contemplated and successfully carried through at the time of the original Mondex development. What was lacking above all then, was the belief that a theorem proving task on such a scale was actually within the scope of the tools of the day, within a reasonable timescale.

3.2 RAISE and PVS at Macao and DTU

Chris George from UNU and Anne Haxthausen from DTU used the RAISE method and associated RSL specification language [RAISE (1995)] to examine the Mondex proof, verifying the RSL text by translation into the input language of the PVS theorem prover [Owre et al. (1995)]. While starting from a fairly faithful representation of the original, they rapidly found that it led to inconvenient and unidiomatic RSL. So they decided to develop their own versions of the models. This classes the work as technology driven. The price to be paid for this is that the delicate invariants, painstakingly discovered by the original developers, no longer worked, and new ones had to be constructed, a time consuming process. Also, the discovery of the tactics to guide PVS in the efficient discharge of various proofs, turned out to be another significant challenge.

3.3 Pefect Developer at Escher Technologies

Perfect Developer from Escher Technologies [Escher], is a modern tool that allows the user to specify his desired system at a high level in an object oriented style, after which an automatic refinement phase can be invoked to turn the specification into working code. The automatic refinement can be replaced by a hand crafted one, after which the tool attempts to prove it correct automatically. Parts of the proof that do not go through automatically are left for the user to handle by interacting with the tool.

One technical snag that that Escher's David Crocker hit quite quickly, was that the original Mondex proof uses 'backward simulation refinement' whereas Perfect Developer is designed for 'forward simulation refinement' (see [de Roever and Engelhardt (1998)] for a discussion of the difference between the two). [2] The Mondex specification was therefore reformulated in a way that was closer to the distributed nature of the actual protocol, after which, this evidently technology driven project was completed in about 60 hours work.

3.4 ASMs and KIV at Augsburg

Gerhard Schellhorn and the Augsburg team came to the Mondex challenge indpendently, and started rather later than other groups. Nevertheless they completed the verification of the entire inter-purse protocol at the end of January 2006, after a month's work, and were the first group to complete the whole challenge. The scope of their work is considerable. They formalised the theory of backward simulation used in the original Mondex development within the KIV theorem prover [KIV], and showed that it constituted a valid notion of refinement. They then formulated the abstract and concrete Mondex models using the ASM formalism, and showed that these models constituted a valid instance of the backward simulation theory, all within KIV [Schellhorn et al. (2006)]. It has to be said that Gerhard and his team have had significant experience of formalising different kinds of refinement notion in KIV, and of proving system developments to be instances of such refinements, but this fact in no way detracts from the scale of their achievement.

The team have built upon their initial success. More recently they have taken the forward simulation approach to Mondex advocated in [Banach et al. (2007)], and have developed it into a fully mechanically verified generalised forward simulation treatment of Mondex [Schellhorn et al. (2007)].

4 After Mondex

The Mondex verification pilot project showed that the verification community is appreciative of the benefits that technical confluence brings, and finds them attractive and convincing. The cross-fertilisation between projects that has been seen in the Mondex pilot is a foretaste of the wider common vision that the VGC hopes to foster. Thus

[2] Following a recent re-examination of the Mondex protocol motivated by different considerations [Banach et al. (2007)], the prospects for a forward refinement between the Mondex abstract and concrete models have now been very much improved.

the VGC envisages following up the Mondex case study with a series of projects of gradually greater scope and complexity.

Candidate projects for such a followup role share a number of characteristics that makes them attractive to the verification community. First and foremeost, they must be of sufficient size that it is evident that customary software development techniques will not guarantee their correctness. Secondly, they must nevertheless be of a size within the reach of today's tools, given reasonable investment in manpower. So a system of several tens of millions of lines of code would be too large to be practicable, even though one might be able to check such a system for specific properties using a tailor-made static checker (as happens these days for Windows^{TM}). Within broad limits, a system of around a hundred thousand lines of code is in the right ballpark. Thirdly, there must be adequate documentation for the system available in the public domain, so that all who wish to tackle such a project can share the same starting conditions.

Beyond these points, if a system has a wider impact than that of simply being a verification exercise, then so much the better. The ideal would be that a fully verified implementation would be created, and that this would then become the 'prefered implementation' worldwide. The more such reference implementations that could be created, the more powerful would the message be that the VGC proclaimed to potentially skeptical observers.

5 A Verifiable File Store

A proposal for a first successor to Mondex comes from NASA's Jet Propulsion Laboratory. Rajeev Joshi and Gerald Holzmann have suggested the creation of a verifiable file store. This is a good suggestion for a number of reasons.

File systems are ubiquitous within the world of computers, and since —independently implemented backup measures aside— all computer data is entrusted to them, the importance of their correct working can hardly be underestimated. Widely used file systems still contain serious bugs that can result in the loss of the entire filestore contents. There is much public domain documentation available on file systems, thus putting such a project within the scope of many teams — the Posix standard constitutes a widely known reference point for many systems. Also there are many approaches that one could contemplate for the verifiable file store. One could start from scratch, drawing up a formal definition of the file system starting from the informal (but quite precise) Posix standard, and then develop this specification to code in a verifiable way. Alternatively, one could take an existing implementation, and attempt to prove it (or a close variant derived as a result of the verification process), correct with respect to a suitably drawn up specification.

An interesting aspect of file system correctness goes beyond mere functional correctness, i.e. beyond whether the file system's data structures and code can store and retrieve data without running into logical flaws. File systems interact with physical media, such as magnetic and optical media, and these days increasingly, flash memory. Each of these has its own individual physical characteristics; beyond which, they all share the possibility of power failure. To be acceptable to users in terms of not experiencing unexpected unrecoverable failures, a file system must take such properties into consideration, and where necessary, modify its interactions with the physical media so as to remain within appropriate parameters.

Flash memory is particularly interesting in this respect. It typically has an upper limit on block usage (say 100,000 uses), blocks can unexpectedly become unusable, bits can flip at random, etc. Developing a file system that is built on such physically shaky foundations, and yet is sufficiently dependable over a reasonably long period, poses significant and interesting challenges. The problems are compounded when you consider that critical software can typically only allocate memory during initialisation, in order to enable static analysis techniques to verify properties of the code in a sufficiently asssured manner. NASA has a particular motivation for seeing the development of such a flash-based filestore, because of the great attractiveness of using non-moving bulk data storage on space missions.[3]

6 Conclusions

We have known for a long time that much software is not as reliable as products in other walks of life. One factor in this is that software is easy to create —personal computers are now relatively cheap, and bits themselves cost nothing, so ‘anyone’ can set about creating software— whereas normal manufactured goods need specialist processes run by trained personnel. Another factor is that a large piece of software has a high complexity, and the nature of its basic ingredients is very brittle —we all know that an unfortunate error in a single bit can have catastrophic consequences for the whole package— whereas in other walks of life, there is much more scope for ‘graceful degradation’.

The potentially anarchic capabilities of a single inappropriately set bit mean that rather draconian measures are needed if one is to be sure that software performs reliably. These go beyond traditional notions of ‘correctness’ for self contained programs, since it is often a failure to appreciate some poorly understood interaction with the environment or interaction between components or subsystems that is the root cause of the bit having become inappropriately set in the first place. (One can mention the famous Mars Polar Lander disaster [Mars Polar Lander] as being a case in point. Poor integration of subsystems meant that the jolt of deployment of a landing leg erroneously set a bit signalling landing, causing the engines to be switched off too early.)

The VGC has all of this in its remit, and with this breadth in mind, is keen to welcome new participants that share its vision and would wish to contribute to its goals. In this paper we have described the first few steps along the road, and these certainly bode well for the future. And whereas in the past there was much reluctance to adopt verification tachniques within the ‘mainstream,’ the more powerful tools that are being created nowadays are much more persuasive that in future, verification tachniques will add genuine value to mainsteam software processes.

[3] In space, spinning up a disk transfers angular momentum from the body of the spacecraft to the disk. This causes the body of the spacecraft to rotate in the opposite direction, potentially causing the spacecraft to lose its orientation in the sky if the rotation is not compensated for by firing correction rockets just the right amount. Spinning the disk down causes the converse problem. In space, accessing data is just as much a problem of mechanics as of computer science.

References

[Alloy] *Alloy Homepage*. `http://alloy.mit.edu/`.
[Escher] *Escher Technologies*. `http://www.eschertech.com`.
[KIV] *The Karlsruhe Interactive Verifier*.
`http://i11www.iti.uni- karlsruhe.de/~kiv/KIV-KA.html`.
[Mars Polar Lander] *Mars Polar Lander Disaster*.
`http://news.bbc.co.uk/2/hi/ science/nature/4522291.stm`.
[OCL (2003)] *OCL Definition*.
`http://www.omg.org/docs/ptc/03-10-14.pdf`.
[QPQ-DSR] *QPQ Deductive Software Repository*.
`http://qpq.csl.sri.com`.
[VST (2005)] *Verified Software: Theories, Tools, Experiments Conference*.
`http://vstte.ethz.ch`.
[Abrial (2007)] J.-R. Abrial. *Event-B*. to be published.
[Banach et al. (2007)] R. Banach, M. Poppleton, C. Jeske, and S. Stepney. Retrenching the Purse: The Balance Enquiry Quandary, and Generalised and (1,1) Forward Refinements. *Fund. Inf.*, 77:29–69, 2007.
[Börger and Stärk (2003)] E. Börger and R.F. Stärk. *Abstract State Machines. A Method for High Level System Design and Analysis*. Springer, 2003.
[Cooper et al. (2002)] D. Cooper, S. Stepney, and J. Woodcock. Derivation of Z Refinement Proof Rules. Technical Report YCS-2002-347, University of York, 2002.
[de Roever and Engelhardt (1998)] W.-P. de Roever and K. Engelhardt. *Data Refinement: Model-Oriented Proof Methods and their Comparison*. Cambridge University Press, 1998.
[ITSEC (1991)] Department of Trade and Industry. *Information Technology Security Evaluation Criteria*, 1991. `http://www.cesg.gov.uk/site/iacs/itsec/media/formal-docs/Itsec.pdf`.
[Common Criteria (2005)] ISO 15408, v. 3.0 rev. 2. *Common Criteria for Information Security Evaluation*, 2005.
[ISO-Z (2002)] ISO/IEC 13568. *Information Technology – Z Formal Specification Notation – Syntax, Type System and Semantics: International Standard*, 2002.
`http://www.iso.org/iso/en/ittf/PubliclyAvailable Standards/c021573_ISO_IEC_13568_2002(E).zip`.
[Owre et al. (1995)] S. Owre, J. Rushby, N. Shankar, and F. von Henke. Formal verification for fault-tolerant architectures: Prolegomena to the design of PVS. *IEEE Transactions on Software Engineering*, 21(2):107–125, February 1995.
[RAISE (1995)] RAISE Method Group. *The RAISE Method Manual*. Prentice Hall, 1995.
[Schellhorn et al. (2007)] G. Schellhorn, H. Grandy, D. Haneberg, N. Moebius, and W. Reif. A Systematic Verification Approach for Mondex Electronic Purses using ASMs. In *Proc. Dagstuhl Workshop on Rigorous Methods for Software Construction and Analysis*, LNCS Festschrift. Springer, 2007.
[Schellhorn et al. (2006)] G. Schellhorn, H. Grandy, D. Haneberg, and W. Reif. The Mondex Challenge: Machine Checked Proofs for an Electronic Purse. In J. Misra, T. Nipkow, and E. Sekerinski, editors, *Proc. FM 2006*, volume 4085 of *LNCS*, pages 16–31. Springer, 2006.
[Spivey (1992)] J.M. Spivey. *The Z Notation: A Reference Manual*. Prentice-Hall, second edition, 1992.
[Stepney et al. (2000)] S. Stepney, D. Cooper, and J. Woodcock. An Electronic Purse: Specification, Refinement and Proof. Technical Report PRG-126, Oxford University Computing Laboratory, 2000.