Under consideration for publication in Formal Aspects of Computing

The Verified Software Repository: a step towards the verifying compiler

J. C. Bicarregui,[1] C. A. R. Hoare,[2] and J. C. P. Woodcock[3]

[1] Rutherford Appleton Laboratory, Chilton, Didcot, Oxon OX11 0QX, UK
[2] Microsoft Research, Roger Needham Building, 7 J. J. Thomson Avenue, Cambridge CB3 0FB, UK
[3] Department of Computer Science, University of York, Heslington, York YO10 5DD, UK

Abstract. The *Verified Software Repository* is dedicated to a long-term vision of a future in which all computer systems justify the trust that Society increasingly places in them. This will be accompanied by a substantial reduction in the current high costs of programming error, incurred during the design, development, testing, installation, maintenance, evolution, and retirement of computer software. An important technical contribution to this vision will be a *Verifying Compiler:* a tool that automatically proves that a program will always meet its requirements, insofar as these have been formalised, without even needing to run it. This has been a challenge for computing research for over thirty years, but the current state of the art now gives grounds for hope that it may be implemented in the foreseeable future. Achievement of the overall vision will depend also on continued progress of research into dependability and software evolution, as envisaged by the UKCRC Grand Challenge project in *Dependable Systems Evolution.*

The Verified Software Repository is a first step towards the realisation of this long-term vision. It will maintain and develop an evolving collection of state-of-the art tools, together with a representative portfolio of real programs and specifications on which to test, evaluate and develop the tools. It will contribute initially to the inter-working of tools, and eventually to their integration. It will promote transfer of the relevant technology to industrial tools and into software engineering practice in the UK. The Repository will build on the recognised achievements of the UK in practical formal development of safety-critical computer applications, and it will establish the UK as the leading nation in any future international initiative in Verified Software, covering theory, tools, and experimental validation.

Keywords: Grand Challenges in Computer Science; *Dependable Systems Evolution*; software engineering; verifying compiler; verified software repository; program verification; assertional reasoning; tools.

1. Introduction

A scientific repository is a selected collection of scientific data constantly accumulating from experimental or observational sources, together with an evolving library of programs that process the data. This paper

Correspondence and offprint requests to: J. C. P. Woodcock, Department of Computer Science, University of York, Heslington, York YO10 5DD, UK. e-mail: `jim@cs.york.ac.uk`.

presents the steps towards setting up a scientific repository serving the advancement of Computer Science, particularly in the area of software engineering and the mechanical certification of computer programs.

The long-term goal of the repository is to assemble, develop, and integrate a of software engineering tool-set for use in the construction, deployment, and continuous evolution of dependable computer systems, and to prove the capability of these tools against a representative portfolio of real computer applications. Establishing this repository will be a substantial step towards development of a standard Verifying Compiler [Hoa03], an outstanding Grand Challenge in Computer Science that goes back more than thirty years.

In Section 2, we describe the background to our proposal: the UK's initiative in grand challenges for computing research, the nature of a verifying compiler, the role of repositories in facilitating scientific research, and the constituency served by our repository. The contents of the Verified Software Repository are described in more detail in Section 3, where we discuss how they will be used in scientific experiments. Finally, in Section 4 we set out our current plans for establishing the repository.

2. Background

2.1. Grand Challenges

The idea of grand challenges has a long history. From the problem of longitude in the 18th Century, through Hilbert's programme for 20th-Century mathematics, to the space race of the 1960s, and the human genome project of the 1990s, grand challenges have led to considerable innovation and have accelerated engineering and technological advancement towards their goal.

The *Human Genome Project* was a 13-year international collaboration on a grand challenge to achieve the following goals [HGP05]:

- *identify* all 25,000 genes in human DNA;
- *determine* the sequences of the three billion chemical base pairs that make up human DNA;
- *store* this information in repositories;
- *improve* tools for data analysis;
- *transfer* related technologies to the private sector; and
- *address* the ethical, legal, and social issues that arise from the project.

The project was completed in 2003, but analysis of the results will continue for many years. The project's success has catalysed the biotechnology industry and fostered the development of new medical applications.

An interesting example of a chain reaction that can be created by a well-articulated grand challenge was the race of autonomous robotic vehicles from Los Angeles to Las Vegas in 2004 [DAR05]; it was organised by DARPA, which offered a $1M prize for the winner. Although none of the entrants came anywhere close to completing the 150-mile obstacle course, and so the prize money was not claimed, the challenge did resulted in a huge engineering effort and fostered significant innovation that might have otherwise been delayed.

In computing, since the early 1990s, NASA has run a series of "science teams" investigating computational grand challenges covering Geophysics, Astrophysics, and Cosmology [NAS05]. Similarly, computational grand challenges have been identified in Astrophysics [IPA05], Molecular Science [MOL05], Aerospace [AGC05], Energy [ENE05], Environmental Monitoring and Prediction [ENV05], Molecular Biology and Biomedical Imaging [MBB05], Product Design and Process Optimisation [PDP05], and Space Science [SSG05]. Grand Challenges in Artificial Intelligence have also been suggested. Most famous of all is the Turing test [TUR05]; more recently, there has been a proposal to extract an encyclopaedic knowledge base from information available on the World-Wide-Web [SWG05]. Our Grand Challenge is the first proposal for the application of computational science to one of the core topics of computer science itself.

Like many research projects and initiatives in Computer Science, these challenges have been motivated and justified by the benefits conferred by computer use on other branches of science, or by its contribution to the needs of Society. A Grand Challenge achieves its rapid effect by opening up new directions of research, and spreading the use of new and promising methods. A Grand Challenge encourages scientific co-operation among disparate teams, and stimulates effective scientific competition. In these ways, a Grand Challenge leads to a significant change in the conduct of the scientists themselves, and thereby significantly accelerates the progress of Science itself. In addition to more tangible benefits, it often finds answers to questions of fundamental intellectual interest at the very basis of the subject, or some significant branch of it.

The UK Computing Research Committee is a body of senior Computer Scientists that has taken upon itself to promote the good health of Computing Research, primarily in the UK, with the aim of making an outstanding contribution to the best of international science. In 2000, it undertook an exercise to identify areas of research in Computer Science that might be significantly promoted by the mechanism of a long-term Grand Challenge. A widespread consultation with the research community revealed seven areas within which a project on the scale of a Grand Challenge, although risky, might bring great rewards. The criteria for a grand challenge and a description of each project may be found in [HoM04, HoM05]. The seven proposed Grand Challenges in Computer Science are:

GC-1 In Vivo $\leftrightarrow$ In Silico [GC105].
GC-2 Science for Global Ubiquitous Computing [GC205].
GC-3 Memories for Life [GC305].
GC-4 Scalable Ubiquitous Computing Systems [GC405].
GC-5 The Architecture of Brain and Mind [GC505].
GC-6 Dependable Systems Evolution [GC605].
GC-7 Journeys in Non-Classical Computation [GC705].

An over-arching committee, GC-0, co-ordinates the work of the individual committees looking after each challenge [GC005]. This paper describes an initiative under the *Verifying Compiler* strand of work in the *Dependable Systems Evolution Challenge*, GC-6.

2.2. The Verifying Compiler

The ideas behind the Verifying Compiler have a long history: it was in 1949 that Turing first proposed using assertions (including loop invariants) in reasoning about program correctness [Tur49]; in 1967, Floyd first proposed the idea of a Verifying Compiler [Flo67]; and in 1968, Dijkstra [Dij68] proposed writing the assertions even before writing the actual program. A *Verifying Compiler* is a tool that proves automatically that a program is correct before allowing it to run. Program correctness is defined by placing assertions at strategic points in the program text, particularly at the interfaces between its components. These assertions are truth-valued expressions that could in principle be evaluated when control reaches a specific point in a program. If the assertion evaluates to false, then the program is incorrect; if it evaluates to true, then no error has been detected. If it can be proved that an assertion will always evaluate to true, then the program is correct, with respect to this assertion, for all possible executions.

Early attempts at constructing a Verifying Compiler were frustrated by the inherent difficulties of automatic theorem proving. These difficulties have inspired productive research on a number of projects, and with the massive increase in computer power and capacity, considerable progress has now been made. Adapting, extending, and tuning existing proof tools to meet the needs of a significant application task would accelerate further progress. We propose *program verification* as just such a task. To this end, we propose to construct a representative collection of real computer software, including the program codes, their specification and other documentation, and suggest them as open challenges for the evaluation and evolution of program verification tools and tool-sets.

2.3. Scientific Repositories

Our proposed first step in a community effort to develop a Verifying Compiler is to set up a *Scientific Repository* in the area of software engineering and the mechanical certification of computer programs. The repository will consist of software analysis tools and example developments to act as challenges for those tools. To apply a tool to a challenge is like a scientific experiment, with its results reported in the literature and added to the repository for use in subsequent experiments.

There are many examples of scientific repositories in the UK, some going back several decades. For example, in molecular chemistry, the *Community Computational Projects* [CCP05] assist with the provision of hardware and software and the definition of standards for inter-operability of programs. The *Cambridge Structure Database* [CSD05] is a repository of crystallographic information for organic molecules. The *NIST Chemistry Web-book* [WEB05] contains data on thermal and spectral properties. Notably the CSD System also includes analysis software for 3-D structure visualisation and for statistics and numerical analysis.

In 1989, the *GDB Database* [GDB05] was set up as an international scientific repository for human gene mapping information; since then, it has proved to be an extremely effective tool for researchers, providing the most up-to-date experimental results and analysis tools for researchers to locate and identify the function of human genes. The repository provided the scaffolding upon which to organise the data of the human genome, and a rough draft of the entire sequence was published in June 2000. This rough draft then became the framework for further advances towards the high-quality finished sequence that we see today. Even now, the work of the repository is far from finished: new gene functions are being elucidated and subjected to peer review, and so the repository is becoming more richly populated with knowledge on the human genome. As well as providing the setting for accessing ever-improving biological data, the repository also provided the setting for improving analysis tools. For example, tools exist to generate and display pair-wise genome alignments. At the beginning of the project, it took over a year of execution on a dedicated processor to produce such an alignment; at the end of the project, this was reduced to less than a week.

In Software Engineering, SourceForge [SFo05] provides a number of services and tools to support the development and distribution of open-source software, and hosts a substantial body of projects in a large number of application areas. Eclipse [ECL05] provides an open platform for tool integration based on plug-ins. The Open Middleware Infrastructure Institute [OMI05a] provides a co-ordinated repository of middleware components for which they have produced a roadmap [OMI05b]. In Formal Methods, the Community Z Tools project [CZT05] has produced open-source libraries for building and integrating Z tools. The Software Engineering Institute at CMU provides a forum for the contribution and exchange of information concerning software engineering improvement activities [SEI05]. QPQ is a repository in the form of an on-line journal for publishing peer-reviewed source code for deductive software components [QPQ05]. Its purpose is to contribute to the scientific infrastructure, and its organisers claim that refereed publication and distribution of software components in open-source form yields higher quality, greater visibility, enhanced interoperability, and accelerated productivity. The *Electronic Tool Integration* platform [ETI05] is a refereed, interactive tool repository accessible as an online resource. Visitors may experiment with individual tools using benchmarks as well as their own examples.

Our proposed Verified Software Repository is very specialised to the needs of a particular research community and research goals, and we propose to referee, evaluate, and co-ordinate submissions in pursuit of these goals. Where appropriate, we will establish links with these other software engineering repositories to reach out to their communities.

An early task in establishing any successful repository is to create a satisfactory IPR policy for its contents. The repository must be open for public browsing, but if users want to run any of the tools or use any of the data, then they will need a licence agreeing that the results of all their experiments will be returned to the repository. The Verified Software Repository's Committee of Guidance will create the policy after consultation with the initial constituency of users.

2.4. The Constituency

The constituency served by the repository will initially consist mainly of researchers engaged in the initial stages of the Grand Challenge, particularly those who have submitted the challenge codes and the analysis programs. But as the scope and maturity of the repository grows, so will the community it serves. Computer scientists from around the world will develop the repository by running the programs on selected data, by interpreting and reporting the results, and by submitting the results to the repository for subsequent use in other investigations. As the project progresses and the utility of the repository grows, the constituency will expand to include leading-edge software development organisations wishing to understand and take advantage of its results to improve the quality of their product. From there, usage will spread into the mainstream of software development in companies looking for assurances of reliability that can be offered by a Verifying Compiler in order to address the widespread and well-justified reluctance in Society to place trust in software [CCP04].

3. The Verified Software Repository

The Verified Software Repository will assist in the development of software by facilitating access to a managed collection of *analysis tools* and a repository of case studies or *challenge codes* to exercise these tools. The

emphasis will be on flexibility: the potential to encompass a broad range of analysis techniques, such as verifiers, theorem provers, model checkers, static analysers, test case generators, *etc* and a broad range of design artefacts, such as documents, codes, test suites, safety cases, *etc*. As a result, the repository will make it easier for software engineers to learn how to use analysis tools effectively. It will also provide examples of good practice for software engineers and facilitate further development and improvement of both the tools and the examples, by bringing them together with common standards.

The long-term goal of the repository, which may take more than a decade to achieve, would be to develop a tool-set of software engineering aids for use in the construction, deployment, and continuous evolution of dependable computer systems. It would provide an initial step and continuing technical support for a proposed Grand Challenge project in Dependable Systems Evolution [UKC05], which exploits and develops the particular strengths of UK computing research. Perhaps such a tool-set would bring about a worthwhile reduction of the high costs of errors in programs, both errors detected before deployment and those detected after. When the tools have proved their maturity, they may be integrated into a standard Verifying Compiler, and so meet the outstanding Grand Challenge in Computing that goes back more than thirty years.

3.1. The Tools

The analysis tools admitted to the repository will include tools that analyse the code for conformity to coding standards, for data flow, for control flow, for alias checking, for type consistency with familiar and novel type systems, for verification condition generation, *etc*. For each language there will be a compiler that makes available an agreed internal interface for inspection and modification by other tools. There will be programs that transform the code, optimise it, and that compare it with previous versions. There will be programs that infer assertions, guess and check conformity to design patterns, generate test data, construct test harnesses, analyse the results of tests, *etc*. Any of these may call on the services of a range of theorem provers, model checkers, constraint solvers, decision procedures, *etc*. In suitable cases there will be competing programs to provide these services, provided that they conform to interfaces that promote interoperability and facilitate direct comparisons of performance.

Our initial list of tools for consideration includes the following: the Daikon dynamic invariant detector [Ern01]; SLAM [Bal01]; the Splint static analyser [Eva02]; the Alloy model checker [Jac02]; the FDR [FSE03] and SPIN [Hol04] model checkers; the ACL2 [KaM96], HOL [Gor88], ProofPower [LeO05], PVS [STF01], and Z/Eves [Saa97] theorem provers; ESC [LNS00]; the Soot Java optimisation framework [VaR99]; the B-toolkit [Abr96]; JML [Lea03]; ASM Engine [ASM05]; SPARK Ada [Bar03]; and the Cyclone safe dialect of the C programming language [JMG02].

3.2. The Challenges

The scientific data held in the repository, the *challenge codes*, will be of varying size and expressed in varying languages. They will be selected because they have proved their significance by actual use in past or present applications. Early examples may be selected from embedded applications (*e.g.*, smart-cards and sensors); from critical control systems (reactors, flood gates); and from open sources of off-the-shelf software (*e.g.*, the Apache web-server). All available forms of machine-readable documentation—specifications, design trajectories, development histories, internal interface specifications, simulators, test case generators, regression tests, and even conjectures, theorems, and proofs—will accompany the codes. Selection of challenge codes will be influenced by availability of this material, and an early task of the analysis programs will be, under human guidance, to regenerate parts of it that are missing (assertions, specifications, design patterns, and even sometimes perhaps the code itself). It is expected that the target levels of certification will be appropriate to the criticality of the application, from basic soundness (crash-proofing), through levels of security, immunity to attack, continuity of service, observance of safety constraints, and ultimately, total conformance to functional specification.

The following well-known experimental applications may be included: the steam boiler [ABL96], the production cell [PRO05]; the storm-surge barrier control [Cha99], the Mondex smart-card [SCW00], railway signalling problems [FME05], the Paris Metro [Car92], the IBM CICS Z specification [HoK91], the inmos floating-point transputer [Bar89], the DeCCo high integrity compiler [Ste98], and ammunition control systems [MuS93]. More ambitious examples may include verifying selected open-software web-services for their

essential functionality, and for crash-freedom and deadlock/livelock freedom. This might lead to verifying the fundamental software building blocks of the whole Internet, with respect to their key properties for their serviceability.

A framework will be devised for evaluating the effectiveness of different approaches and tools when applied to the challenge codes. This will be done in collaboration with certification agencies. An example of such a framework is the one being devised by the *HIRTS DARP (Defence and Aerospace Research Partnership in High-Integity Real-Time Systems)* [DAH05].

Each year, prizes will be awarded for the best contributions to the work of the repository. The prizes will be administered by an independent committee of international experts in association with learned and professional societies. Some industrial collaborators have already offered to sponsor this activity.

3.3. Dependable Systems Evolution

Many of the challenge codes will be undergoing evolution at the hands of their owners and users, and it will be part of the challenge to track the evolution. This will involve further research into the scientific challenges posed by systems evolution, as well as sophisticated configuration and change management disciplines. If the challenge is met, the owners will be pleased at some stage to take over the annotated code, and use the tools to assist in its further evolution. Of course, the tools will only address aspects of the process of evolution that can be formally expressed. The Challenge will be to widen as far as possible that range, and to collaborate with investigations into the wider aspects of evolution of dependable systems that would not benefit from formalisation. Similarly, the concept of dependability has many important aspects that cannot yet be formalised, and many that never will be. The project will aim to extend the range of formalisation, and collaborate with centres researching the wider problem, using a wider range of scientific techniques than mathematical modelling and formalisation.

4. Current Plans

We now describe the specific objectives, proposed activities, and work-plan for a co-ordinated community project to establish the repository described above. We begin by giving the goals of the project.

4.1. Goals

1. To establish and maintain UK leadership in a long-term worldwide Grand Challenge project, specifically: *Verified Software: Theories, Tools, and Experiments* [VST05].
2. To promote scientific collaboration and constructive rivalry amongst researchers engaged in the project.
3. To provide educational, advisory, administrative, public relations, and programming support for UK and European participants.
4. To communicate and collaborate with similar repositories in other parts of the world.
5. To assist in the transfer of maturing technology to industrial and commercial exploitation.

4.2. Objectives

To accelerate the advancement of technology contributing to the development of the *Verifying Compiler* by providing a repository of tools and challenges for software analysis and hence facilitate progress towards the Grand Challenge in *Dependable Systems Evolution*. Specifically,

1. To facilitate the widespread and effective use of software analysis tools by providing access to a managed collection of tools such as verifiers, provers, model checkers, test-case generators, test rigs, *etc.*
2. To improve understanding of design issues in enabling software verification through provision of a managed repository of case studies demonstrating good practice.
3. To encourage further development and improvement of both the tools and the challenges by bringing

them together with common standards to enable the example developments to be used to improve the design of the software analysis tools

4. To provide resources to support the repository users in order to encourage its widespread use and hence maximise the benefits arising from its provision.

4.3. Key outcomes over one, five, and ten years

Year 1

Milestones

1. Establish repository and supporting resources and services: recruit and train staff, install hardware and software, design and implement website, set up helpdesk.
2. Recruit and assemble an initial constituency of users.
3. Establish and agree terms of reference for the *User Forum*.
4. Set up and agree terms of reference for the *Committee of Guidance*.
5. Populate the repository with case studies contributed by the user community.
6. Populate the repository with research tools selected from existing UK and European sources.
7. Populate the repository with major challenge codes selected from existing critical applications.

Results

1. To have one tool available in the repository for each of the tasks that face us:
   - Development of code from specifications: *e.g.*, the B-tool [Abr96].
   - Development of assertions from code: *e.g.*, TVLA for abstract interpretation [TVL05].
   - Development of assertions from tests: *e.g.*, the Daikon dynamic invariant detector [Ern01].
   - Verification condition generation: *e.g.*, ESC/Java [LNS00].
   - Theorem proving: *e.g.*, PROSPER (proof and specification-assisted design environments) [PRO05].
2. To have agreed and installed initial case studies and major challenge codes.
3. To have identified a community of software engineering researchers wishing to use the repository.

The examples in (1) are given merely as an existence proof that are mature tools that fill each slot. Furthermore, there are grounds for hope that this particular tool-set could be readily adapted to work together in experiments conducted with Java. Many of the tools originate in Europe, so their further development is natural as a European responsibility. It is to be hoped that more local repositories will take up responsibility for tools originating in other continents, and make them freely available worldwide, mediated by the local repositories. Tools originating in the USA are listed at [VGC05]. Further discussion is essential before any decisions are reached on the selection of tools.

An academically respectable review process will be set up so that submission can garner academic credit.

Year 5

Milestones

1. Support provided for an extended tool-set, and for further development of existing tools.
2. Support provided for a wider range of challenge codes; existing challenge codes will have been subjected to experimental annotation and proof, with the results recorded in the repository.
3. Substantial inter-operability between tools and challenges: each code being compatible with several tools, and each tool applicable to several codes.
4. Agreed standards published for universal inter-working of tools and challenge codes.
5. Trial integration of a selected group of tools.

Results

- An active community of academic and industrial repository users depositing and using the repository.
- A significant library of software development examples to assist with education and training in software engineering techniques.
- Some examples of community development of case studies by interested parties applying their tools and techniques to library challenges.
- Significantly increased inter-operability of tools through standardisation of interfaces.

Year 10

Milestones

1. An integrated tool-set of mature tools in general use by researchers.
2. Almost full inter-working of tools and challenge codes.
3. Fifty challenge codes, up to a million lines in length.
4. Industrial and commercial trials underway.
5. Some parts of the technology transferred to commercial tool-sets.
6. Established library of twenty software systems, mechanically verified to an appropriate and formally defined level of correctness, and beyond.

Results

- Availability of one-click analysis tools offering some assurance for minimal investment.
- Regular examples of community development of case studies by interested parties applying their tools and techniques to library challenges.
- A substantial community of repository users actively depositing and exploiting the tools and challenges in the repository.

4.4. Activities

The project will undertake a number of activities and manage a suite of resources on behalf of the software engineering community. Co-ordinated management of these activities and resources will improve overall efficiency by enabling some standardisation and economies of scale. Examples of activities that would be appropriate are listed below. The activities break down into the five work-packages described below.

4.4.1. Tools

Each year, the Committee of Guidance will select a number of tools for inclusion in the repository and will negotiate with their authors for their acquisition, distribution, and support as appropriate. Support will be provided for dissemination of the tools in order to facilitate their application to example challenges.

- Negotiation with providers of analysis tools for the requisite releases and IPR, together with material to support an advisory role, and a feedback path for correction and improvement of the tool.
- Licence co-ordination and negotiation where appropriate.
- Administer tool upgrades, so as to protect continuity for researchers.

4.4.2. Challenges

Each year, a number of challenge codes will be selected for inclusion in the repository. The CoG will negotiate with the authors of the codes for their acquisition, and then install them in the repository. The CoG will provide support for dissemination updates from the owners of the challenges in order to facilitate their uptake by tool providers.

- Negotiation with providers of challenge codes for requisite releases and IPR, together with specifications, test sets, development trajectories, *etc.*
- Copyright co-ordination and negotiation where appropriate.
- Data conversion in order to maximise analysis by repository tools.

4.4.3. Repository

This work-package will set up, manage and develop the repository, which will be organised around a web-based system for depositing and accessing tools, challenges, training materials, publications, and project data. It comprises two parts. In Task 1, we will gather requirements, design, build and maintain the repository and web-based access to it. In Task 2, we will develop standards and supporting software enabling inter-operability between tools and challenges.

- Provision of a repository for use by the community to deposit and access tools and challenges for software analysis.
- Design and implementation of interface converters to ensure that the same (selected) analysis tools will apply to the same (selected) challenge codes. (Some of this work could be contracted out or commissioned).
- Co-ordination of the development of standards to enable inter-working of tools and challenges.
- Negotiation with providers of tools and codes for standards that will facilitate inter-working without specialised converters.
- Provision of aids to standards compliance, monitoring and enforcement.
- Long-term curation of tools and challenges enabling continued use of older versions in order to permit repetition of earlier experiments and assessment of progress.

4.4.4. User Support

This work-package will also be responsible for providing support to the constituency of potential users of the repository. It will provide supporting material, events and a helpdesk for users of the repository.

- Populate, maintain, refine, and extend a web-site framework including:
  - guidance on installation and/or web use of tools
  - guidance of the case studies software
  - training material on use of the repository including that arising from the organised events.
- Helpdesk providing first level support for repository users.
- Support for the publication of newsletters and a journal devoted to publication of the results of verification experiments, before the results are incorporated in the repository.
- Planning and conduct of educational courses and updates for participants from the community.
- Organisation of user forums and events including an annual Marktoberdorf-style summer school devoted to training in the use of that year's tools and in the understanding of that year's challenge codes.

4.4.5. Management and Dissemination

This work-package will co-ordinate the management of the project and will oversee the publicity and public relations surrounding it.

- Establishment and administration of a management structure to pursue the scientific goals of the project and to act as guarantors of its scientific integrity. (See Section 4.5.)
- Assistance in establishment and maintenance of scientific relations with similar repositories in other countries.
- Organisation and hosting for meetings, workshops, conferences, summer schools in relevant topics.
- Establishment and organisation of newsletters, websites, and publications, including an international Journal of record for the results of the research.

4.5. Management

An oversight body, the *Committee of Guidance*, will be established to steer the project and to monitor its use of resources. The Committee of Guidance will review the programme of work and set priorities annually, and review progress and performance metrics against that programme quarterly. The initial Committee of Guidance will review the programme of work for the first year and will establish a *User Forum* and procedures for electing future Committees of Guidance. The Committee will consist originally of the promoters of the project,[1]and subsequently of those elected by the constituency.

We will also establish a *Stakeholder Forum* for users of the facility (tool providers, case study providers, users, *etc*) to give feedback on the operation of the facilities provided and advise the Committee of Guidance. Initially, membership of the user forum will be open, although it may eventually become necessary to limit numbers by election of representatives from various classes of user.[2]

4.5.1. Committee of Guidance

The role of the Committee of Guidance is:

1. To pursue single-mindedly the scientific goals of the project, and act as guarantors of its scientific integrity.
2. To direct the policies of the repository and monitor performance.
3. To lay down appropriate procedures for the selection and classification of material for acceptance into the repository.
4. To assess annually the current status of the repository and draw up prioritised plans for its future progress.
5. To negotiate with its constituency a division of labour to implement the plans.
6. To negotiate issues of inter-working standards and work-plans with its constituency and with related centres in other nations.
7. To lay down and administer fair procedures for entry of new candidates to the constituency.
8. To make commendations of teams who have made exceptional contributions.
9. To arrange for conferences, workshops, seminars, summer schools, meetings, and for publication of newsletters and journals in furtherance of the aims of the project.
10. To promote wider education in the understanding of new tools and their potential use.
11. To promote general public awareness of, and engagement, with the nature of the science and the goals and achievements of the project.

4.5.2. Stakeholder Forum

The Stakeholder Forum will discuss options for the content and function of the repository and advise the Committee of Guidance on its development. The role of the Stakeholder Forum is:

1. To provide a forum for the community of users of the repository to discuss their work.
2. To provide feedback to the CoG on the repository and its associated activities.
3. To advise the CoG on the content and functionality of the repository.
4. To contribute to the organisation of events, publicity, and other materials concerning the repository.
5. To make recommendations to the CoG on strategic directions for the repository.
6. To liaise with other related user forums, such as Formal Methods Europe and the ASM, B, Refinement, TPHOL, and Z user groups.

[1] Suggested initial members of the Committee of Guidance are: Juan Bicarregui (RAL), Tony Hoare (Microsoft), Cliff Jones (Newcastle), Colin O'Halloran (QinetiQ), Peter O'Hearn (Queen Mary), and Jim Woodcock (York).
[2] Suggested initial members of the Stakeholder Forum are: those who have contributed content to the repository and any other interested parties such as representatives of their institutions, representatives of the funders *etc*.

4.6. Resources

Due to necessarily limited resources, the project will have to start relatively small; however, a certain scale of perhaps a dozen tools and a dozen case studies would be the minimum required to establish the facility as a useful addition to the status quo. Once established, we imagine that gathering momentum will encourage contributions from Repository users (*c.f.*, Wikipedia) and so provide significant gearing to the centralised resources. Ultimately we expect a fairly large-scale operation being maintained through relatively little dedicated resource.

The existence of the repository proposed here is expected to have a significant effect on the behaviour of scientists working in the field and so ensure that the results of their collaborative research are complementary and cumulative. If successful, this will have a radical influence on the speed of scientific progress in the area and will bring forward the development of the Verifying Complier and with it the more widespread application of software analysis tools into mainstream software engineering practice.

Acknowledgement

This paper has been prepared in discussion with the GC6 Steering Committee. The members of the Steering Committee are: Keith Bennett, Juan Bicarregui, Jonathan Bowen, Tony Hoare, Cliff Jones, John McDermid, Colin O'Halloran, Peter O'Hearn, Brian Randell, Martyn Thomas, and Jim Woodcock.

References

[ABL96] Abrial, J.-R., Börger, E., and Langmaack, H. (editors): *Formal Methods for Industrial Applications: Specifying and Programming the Steam Boiler Control. Lecture Notes in Computer Science* ***1165***. Springer-Verlag, 1996.

[Abr96] Abrial, J.-R.: *The B-Book: Assigning Programs to Meanings*. Cambridge University Press, 1996.

[AGC05] Aerospace Grand Challenge. Cited 2005.
`ftp://avalon.caltech.edu/pub/murray/preprints/mur03-ejc.pdf`.

[ASM05] *Abstract State Machines: Tools*.
`www.eecs.umich.edu/gasm/`. Cited 2005.

[Bal01] Ball, T., *et al.*: Automatic predicate abstraction of C programs. *ACM SIGPLAN Conference on Programming Language Design and Implementation* 203–213 (2001).

[Bar03] Barnes, J: *High Integrity Software: the* SPARK *Approach to Safety and Security*. Addison-Wesley, 2003.

[Bar89] Barrett, G.: Formal methods applied to a floating-point number system. *IEEE Transactions on Software Engineering* ***15***(5):611–621 (1989).

[Car92] Car92, M., C. Da Silva, Dehbonei, B., and Mejia, F.: Error-free software development for critical systems using the B-methodology. *Third IEEE International Conference on Software Reliability*: 274–281 (1992).

[CCP04] CCP04 and Crime Prevention Project, Department of Trade and Industry, Foresight, June 2004.
`www.dti.gov.uk, DTI/Pub7186/5K/NP`.
`www.foresight.gov.uk/CCP04.html`.

[CCP05] The Collaborative Computational Projects (CCPs)
`www.ccp.ac.uk/`. Cited 2005.

[Cha99] Cha99, M., Tretmans, J., and Wijbrans, K: Lessons from the application of formal methods to the design of a storm surge barrier control system. *FM'99: World Congress on Formal Methods in the Development of Computing Systems. Volume II*. Editors: Wing, J. M., Woodcock, J., and Davies, J. *Lecture Notes in Computer Science* ***1709***:1511–1526 Springer-Verlag (1991).

[CSD05] `www.CSD05.cam.ac.uk/products/csd/`. Cited 2005.

[CZT05] `czt.sourceforge.net/`. Cited 2005.

[DAH05] BAE Systems, Rolls-Royce plc, QinetiQ, and the University of York: *Defence and Aerospace Research Partnership in High-Integrity Real-Time Systems*. Strand 3: Measurement and Management of HIRTS.
`www.cs.york.ac.uk/hise/darp/`. Cited 2005.

[DAR05] The DARPA Grand Challenge for Autonomous Robotic Ground Vehicles,
`www.darpa.mil/grandchallenge04/sponsor_toolkit/brochure.pdf`. Cited 2005.

[Dij68] Dijkstra, E. W.: A Constructive Approach to the Problem of Program Correctness. *BIT* ***8***(3):174–186 (1968).

[ECL05] `www.eclipse.org`. Cited 2005.

[ENE05] Energy Grand Challenge.

[ENV05] Grand Challenges in Earth and Environmental Sciences: Science, Stewardship, and Service for the Twenty-First Century, Zoback, M. L., U.S. Geological Survey,
`www.environmentalsafeguards.com/zoback.pdf`. Cited 2005.

[Ern01] Ernst, M. D., Cockrell, J., Griswold, W. G., and Notkin, D: Dynamically discovering likely program invariants to support program evolution. *IEEE Transactions on Software Engineering* ***27***(2):1–25 (2001).

[ETI05] Electronic Tool Integration Platform. *Software Tools for Technology Transfer*. `sttt.cs.uni-dortmund.de/`. Cited 2005.

[Eva02] D. Evans and Larochelle, D: Improving Security Using Extensible Lightweight Static Analysis, *IEEE Software* (2002).

[Flo67] Floyd, R. W.: Assigning meanings to programs, *Proc. Amer. Soc. Symp. Appl. Math.* ***19***:19–31 (1967).

[FME05] *FMERail: Workshops on Formal Methods in the Railway Industry*. `www.ifad.dk/Projects/fmerail.htm`. Cited 2005.

[FSE03] *Failures-Divergence Refinement: FDR2 User Manual*. Formal Systems (Europe) Ltd 2003. `www.fsel.com`.

[GC005] Hoare, C. A. R., Atkinson, M., Bundy, A., Crowcroft, J., McDermid, J. M., Milner, R., Moore, J., Rodden, T., and Thomas, M.: *The Grand Challenges Exercise of the UKCRC*. `www.nesc.ac.uk/esi/events/Grand_Challenges/PC-report.pdf`. Cited 2005.

[GC105] *In Vivo ↔ In Silico*. `www.nesc.ac.uk/esi/events/Grand_Challenges/proposals/ViSoGCWebv2.pdf`. Cited 2005. Contact: R. Sleep.

[GC205] *Science for Global Ubiquitous Computing*. `www.nesc.ac.uk/esi/events/Grand_Challenges/proposals/Ubiq.pdf`. Cited 2005. Contact: M. Kwiatkowska and V. Sassone.

[GC305] *Memories for Life*. `www.nesc.ac.uk/esi/events/Grand_Challenges/proposals/Memories.pdf`. Cited 2005. Contact: A. Fitzgibbon and E. Reiter.

[GC405] *Scalable Ubiquitous Computing Systems*. `www.nesc.ac.uk/esi/events/Grand_Challenges/proposals/Memories.pdf`. Cited 2005. Contact: J. Crowcroft.

[GC505] *The Architecture of Brain and Mind*. `www.nesc.ac.uk/esi/events/Grand_Challenges/proposals/ArchitectureOfBrainAndMind.pdf`. Cited 2005. Contact: A. Sloman.

[GC605] *Dependable Systems Evolution*. `www.nesc.ac.uk/esi/events/Grand_Challenges/proposals/dse.pdf`. Cited 2005. Contact: J. Woodcock.

[GC705] Journeys in Non-Classical Computation, `www.nesc.ac.uk/esi/events/Grand_Challenges/proposals/stepney.pdf`. Cited 2005. Contact: S. A. Stepney.

[GDB05] GDB(TM) Human Genome Database [database online]. Baltimore (Maryland USA): Johns Hopkins University, 1990–. Updated daily. Available from Internet URL `www.gdb.org/`. Cited 2005.

[Gor88] Gordon, M. J. C.: HOL: A proof generating system for Higher-Order Logic. *VLSI Specification, Verification and Synthesis*, Kluwer, 73–128 (1988).

[HGP05] *Human Genome Project Information*. `www.oml.gov/sci/techresources/Human_Genome`. Cited 2005.

[Hoa03] Hoare, C. A. R.: The Verifying Compiler: a Grand Challenge for Computer Research. *JACM* ***50***(1):63–69 (2003).

[HoK91] Houston, I and King, S: CICS project report: Experiences and results from using Z. *VDM'91: Formal Development Methods. Lecture Notes in Computer Science* ***551*** Springer-Verlag, 1991.

[Hol04] Holzman, G. J.:*The SPIN Model Checker: Primer and Reference Manual*. Addison Wesley, 2004.

[HoM04] Hoare, T. and Milner, R.: *Grand Challenges in Computing: Research*. The British Computer Society, Swindon. 2004. `www.ukcrc.org.uk/gcresearch.pdf`.

[HoM05] Hoare, C. A. R., Milner, R.: Grand Challenges for Computing Research. *The Computer Journal* ***48***(1):49–52 (2005).

[IPA05] Institute for Pure and Applied Mathematics, Grand Challenge Problems in Computational Astrophysics. `www.ipam.ucla.edu/programs/pca2005/`. Cited 2005.

[Jac02] Jackson, D: Alloy: A lightweight object modelling notation. *ACM Transactions on Software Engineering and Methodology* ***11***(2):256–290 (2002).

[JMG02] Jim, T., Morrisett, G., Grossman, D., Hicks, M., Cheney, J., and Wang, Y: Cyclone: A safe dialect of C. *USENIX Annual Technical Conference*, Monterey (2002).

[KaM96] M. Kaufmann and Moore, J. S.: ACL2: An industrial strength version of Nqthm. *Compass'96: Eleventh Annual Conference on Computer Assurance* (2002).

[Lea03] Leavens, G. T. *et al.*: An overview of JML tools and applications. ***in*** Arts & Fokkink (editors) *Eighth International Workshop on Formal Methods for Industrial Critical Systems (FMICS'03). Electronic Notes in Theoretical Computer Science* volume ***80***:73–89 (2003). Elsevier.

[LeO05] `www.lemma-one.com/ProofPower/index/`. Cited 2005.

[LNS00] Leino, K. R. M., Nelson, G., and Saxe, J. B.: *ESC/Java User's Manual*. Technical Note 2000-002. Compaq Systems Research Center (2000).

[MBB05] `dcegod.nci.nih.gov/epi/BCBRWG.htm`. Cited 2005.

[MOL05] The Computational Grand Challenge Applications (CGCA) projects in environmental molecular science. `mscf.emsl.pnl.gov/research/intro_cgca.shtml`. Cited 2005.

[MuS93] Mukherjee, P. and Stavridou, V.: The formal specification of safety requirements for storing explosives. *Formal Aspects of Computing* ***5***(4):299–336 (1993).

[NAS05] NASA: Science Team III Grand Challenge Investigators, Computational Technologies for Earth and Space Sciences. ct.gsfc.nasa.gov/grand.st3.html. Cited 2005.
[OMI05a] www.omii.ac.uk/. Cited 2005.
[OMI05b] www.omii.ac.uk/OMII_Roadmap_Feb04.pdf. Cited 2005.
[PDP05] Product Design and Process Optimization Grand Challenge.
[PRO05] PROSPER: Proof and Specification Assisted Environments. www.dcs.gla.ac.uk/prosper. Cited 2005.
[QPQ05] *QED Pro Quo.* www.qpq.org/. Cited 2005.
[Saa97] Saaltink, M.: The Z/Eves System. *Proceedings of the 10th International Conference of Z Users.* Lecture Notes in Computer Science ***1212***:72–85 (1997).
[SCW00] Stepney, S., Cooper. D. and Woodcock, J. C. P.: *An Electronic Purse: Specification, Refinement, and Proof.* ***PRG-126***, Oxford University Computing Laboratory, 2000.
[SEI05] seir.sei.cmu.edu/seir/frames/frmset.help.asp. Cited 2005.
[SFo05] SourceFORGE sourceforge.net/. Cited 2005.
[SSG05] Space Science Grand Challenge.
[Ste98] Stepney, S: Incremental development of a high integrity compiler: Experience from an industrial development 1998. *HASE'98: Third IEEE High-Assurance Systems Engineering Symposium* (1998).
[STF01] Shankar, U., Talwar, K., Foster, J. S., and Wagner, D.: Detecting format string vulnerabilities with type qualifiers. *Proceedings of the 10th USENIX Security Symposium* (2001).
[SWG05] Semantic Web Road map. www.w3.org/DesignIssues/Semantic.html. Cited 2005. See also www.w3.org/DesignIssues/RDFnot.html, www.cs.vu.nl/~ frankh/postscript/IJCAI99-III.html, and www.ht03.org/papers/pdfs/7.pdf.
[Tur49] A. M. Turing. Checking a large routine. *Report of a Conference of High Speed Automatic Calculating Machines* pp.67–69. Univ. Math. Lab. Cambridge. 1949.
[TUR05] The Turing Test. *Stanford Encyclopedia of Philosophy.* plato.stanford.edu/entries/turing-test/. Cited 2005.
[TVL05] TVLA: 3-Valued Logic Analysis Engine. www.cs.tau.ac.il/~tvla/. Cited 2005.
[UKC05] UKCRC 2004. *Dependable Systems Evolution.* www.nesc.ac.uk/esi/events/Grand_Challenges/proposals/dse.pdf. Cited 2005.
[VaR99] Vallée-Rai, R. *et al.*: Soot—a Java optimization framework. *Proceedings of CASCON 1999* 125–135 (1999).
[VGC05] www.csl.sri.com/~shankar/VGC05. Cited 2005.
[VST05] *Verified Software: Theories, Tools, and Experiments.* vstte.ethz.ch/. Cited 2005.
[WEB05] webbook.nist.gov/. Cited 2005.