The FMICS view on the Verified Software Repository

Alvaro Arenas, Juan Bicarregui, Tiziana Margaria

Published version information

Citation: AE Arenas, JC Bicarregui and T Margaria. 'The FMICS view on the Verified Software Repository.' Journal of Integrated Design and Process Science, vol. 10, no. 4 (2006): 47-54.

The final publication is available at IOS Press through:
https://content.iospress.com/articles/journal-of-integrated-design-and-process-science/jid10-4-05

Integrated Design and Process Technology, **IDPT-2006**
Printed in the United States of America, June, 2006

The FMICS View on the Verified Software Repository

Alvaro Arenas[1], **Juan Bicarregui**[1], **Tiziana Margaria**[2]

[1]**CCLRC Rutherford Appleton Laboratory, UK**
{A.E.Arenas, J.C.Bicarregui}@rl.ac.uk

[2]**Chair of Service and Software Engineering**
Universität Potsdam
margaria@cs.uni-potsdam.de

ABSTRACT: An important step in meeting the Verifying Compiler Grand Challenge is the Verified Software Repository. In the FMICS view, the repository should include proven correct software and tools to help establishing the correctness of the software in question. We propose to set up a collaborative demonstrator, based on the jETI technology, to provide tools to the repository and to orchestrate different tools.

I. Introduction

The Verifying Complier Grand Challenge aims at building a verifying compiler that uses mathematical and logical reasoning to check the correctness of the programs that it compiles [9]. As mentioned by Hoare in [9], the compiler will work in combination with other program development and testing tools in order to achieve any desired degree of confidence in the structural soundness of the system and the total correctness of its more critical components.

A step towards the Verifying Compiler is the Verified Software Repository [3], a scientific repository that 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.

This paper presents the views on the ERCIM Working Group on Formal Methods for Industrial Critical Systems (FMICS) [27] on the VSR. One of the goals of FMICS is to transfer and promote the use formal methods technology in industry. The authors believe that the VSR offers a great opportunity to reach this goal, resulting in a more robust and solid software industry in Europe.

II. An Overview on Scientific Repositories

The provision of scientific repositories is one example of facilitating access to research outputs. A good example of the progress that can be facilitated by a scientific repositories is the GDB Database [8] which was set up in 1989 as an international scientific repository for human gene mapping information and provided the scaffolding upon which to organise the data of 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, at the beginning of the project, it took over a year of execution on a dedicated processor to generate and display pair-wise genome alignment; at the end of the project, this was reduced to less than a week.

There are many examples of scientific repositories in Europe, some going back several decades. For example, in molecular chemistry, the UK Community Computational Projects [4] coordinate the provision of hardware and software within particular application areas, and assist in the definition of standards for inter-operability of programs and for their admission to the repository, and provide organisation for visitors, workshops, seminars, summer schools, etc. They are steered by committees representing the whole constituency.

In Software Engineering, a number of initiatives have come out in the last decade, with rather different aims and character:

- SourceForge [13] 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 [6] provides an open platform for tool integration based on plug-ins.
- The Open Middleware Infrastructure Institute [10] provides a co-ordinated repository of middleware components.
- In Formal Methods, the Community Z Tools project [5] 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 [12].
- QPQ is a repository in the form of an on-line journal for publishing peer-reviewed source code for deductive software components [11]. 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 (ETI) [23], [24] is an online platform specifically designed to support the distributed use of and experimentation with tools over the

internet. Born in 1996 and online since early 1997, it offered a unique service to tool providers and users: its solution for the remote execution of tools and the internet-based administration of user-specific home areas on the ETI server was well ahead of the technology of those times. Since then, users can *retrieve information* about the tools, *execute tools* in *stand-alone mode*, or *combine functionalities of different tools* to obtain sequential programs called coordination sequences and run them in tool-coordination mode. In particular, ETI is unique in allowing users to combine functionalities of tools of different application domains to solve problems a single tool never would be able to tackle.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.

However, these efforts are not yet coordinated, neither toward common goals, nor in the way they act and develop.

The great value of Hoare's Verifying Compiler Grand Challenge is in its capability of aligning several of these initiatives into a long term, broadband effort of a large community, leveraging synergies and the exploitation of techniques, methods, and results. De facto, this unites the community at large and fosters the communication across boundaries.

A central aspect of this Challenge is the realization of a repository of verified software. From the point of view of FMICS, the acception of *software* should encompass not only code, but also models, and the repository should contain not only analyzed or proven correct software, but also *tools* (themselves software artifacts) that help establishing the correctness of the software in question starting from the requirements, specifications, and models. Admittedly, this point of view that stresses correctness at the model level is particularly important in the FMICS community, which develops since its inception methods, tools, and their applications to industrial critical systems. However, we are convinced that the extension to cover tools and even frameworks can be useful for the progress towards better software also for less critical application domains, like consumer IT products: they strike wherever *time to market* and *first time correct* are an issue.

III. UK Efforts towards a Verified Software Repository

UK has initiated effort towards a verified software repository [3] within the framework of the UK Grand Challenges Programme [14] throughout the Grand Challenge in Dependable Evolution Systems [16]. 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 analyzers, 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.

By its nature, the value of the repository comes from it being long lived. The repository will provide a tool-set of software engineering aids for use in the construction, deployment, and continuous evolution of dependable computer systems. It would provide a major step and continuing technical support for a proposed Grand Challenge project in Dependable Systems Evolution, which exploits and develops the particular strengths of UK computing research. 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.

A. The Tools

The analysis tools admitted to the repository will include tools that analyse 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 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.

B. 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 of-the-shelf software (e.g., the Apache web-server). All available forms of machine-readable documentation, specifications, design trajectories, development histories, inter-

nal 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.

C. Methodology

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. 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 [7].

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

IV. The FMICS Contribution

Based on jETI [21], the new generation of ETI [23], the core FMICS partners are going to set up a collaborative demonstrator that

- illustrates the applicability of the jETI technology for lightweight remote integration of tools into the repository
- shows how to provide tools to the repository, by registration and remote provision,
- demonstrates how to experiment with local and remote tools to solve cooperative verification tasks
- shows how to orchestrate different tools (possibly a mix of local and remote ones) which were not originally designed to cooperate, to address more complex case studies. This may require the availability of mediators, to cover semantic gaps between the tools.

The participating partners are so far CCLRC/RAL (UK), CWI (The Netherlands), INRIA Grenoble (France), ISTI Pisa (Italy), Masaryk University in Brno (Czech Republic), University of Dortmund (Germany), the University of Malaga (Spain), University of Potsdam (Germany), University of Saarbrücken (Germany).

The core effort by the partners concerns the provision of tools and case studies that serve as executable benchmarks of the demonstrator. The coordinators (Univ. of Dortmund and Univ. of Potsdam) are responsible for the jETI platform and technology, and for the development and test of the demonstration between their two locations.

V. Easy Remote Tool Integration in jETI

ETI's purpose was unique in allowing users to combine functionalities of tools of different application domains to solve problems a single tool never would be able to tackle.

Obviously, the richness of the *tool repository* plays a crucial role in the success of ETI: the benefit gained from our experimentation and coordination facilities grows with the amount and variety of integrated software-tools. The success of the ETI concept is thus highly sensitive to the process and costs of *tool integration* and *tool maintenance*.

Taking advantage of newer technologies that internally base on Web Services and Java technology, we can
1. considerably simplify the integration process, and at the same time
2. flexibilize the distribution, version management and use of integrated tools,
3. broaden the scope of potential user profiles and roles, by seamlessly integrating ETI's coordination and synthesis features (cf. [23]) with a standard Java development environment, and
4. solve the scalability problem connected with tool maintenance and evolution.

The background and a first attempt to the new distributed way of tool integration for ETI have been described in [19]. Our current version of ETI, jETI,

- exploits Web Services technology [30], [29], [28] to further simplify the remote tool integration and execution,
- supports cross platform execution of the coordination models based on the quasi standard set by Java, and it naturally
- flexibilizes the original coordination level by seamlessly integrating the Eclipse development framework [6].

A. Registration instead of Integration

jETI's integration philosophy addresses the major obstacle for a wider adoption, as identified during seven years of experience with tool providers, tool users and students: the difficulty to provide the latest versions of the state-of-the-art tools. The tool integration process required on dedicated ETI servers was too complicated for both the tool providers and the ETI team, making it impossible to keep pace with the development of new versions and a wealth of new tools. jETI's new remote integration philosophy overcomes this problems, because it replaces the requirement of 'physical' tool integration by a very simple registration and publishing. This allows the provision of tool functionalities in a matter of minutes: fast enough to be fully demonstrated during our presentation. Moreover, whenever the portion of a tool's API which is relevant for a new version of a

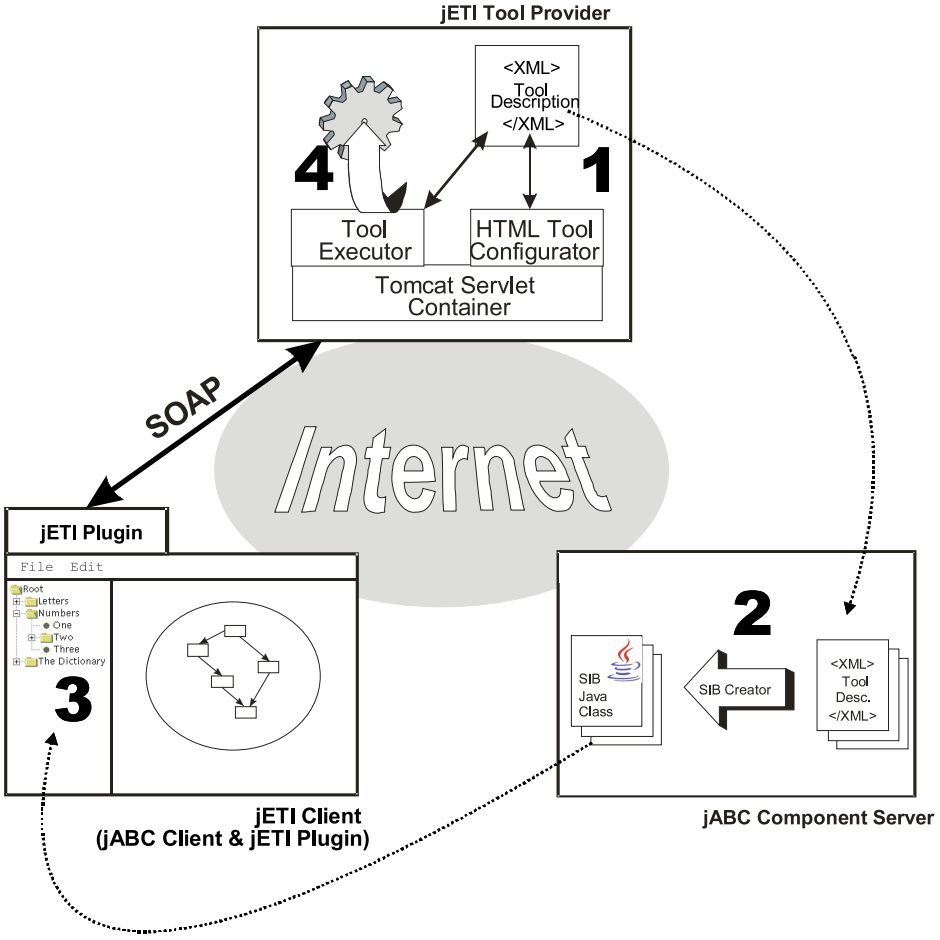

Fig. 1. The jETI Tool Integration Workflow

functionality remains unchanged, version updating is fully automatic!

Based on the Web Services functionality, the realization of this registration/ publishing based integration philosophy required the implementation of four components, as illustrated in Figure 1:

1. a **HTML Tool Configurator**, which allows tool providers to register a new tool functionality just by filling our a simple template form,
2. the **jABC Component Server**, which (a) automatically generates appropriate Java classes from these specifications and (b) organizes all the registered tool functionalities, including the corresponding version control,
3. the **jETI client**, which automatically loads the relevant Java classes from the jABC Component Server and provides a flexible Java development environment for coordinating the so obtained tool functionalities. Depending on their goals and skill profile, users may just use our graphical coordination editor to experiment with the tools, or they may use the full development support of Eclipse to really embed remote functionalities into normal Java programs. Of course, this choice heavily influences the size of the required jETI-Client: only the first option is open to our envisaged 'pure HTML' solution.
4. a **Tool Executor**, which is able to steer the execution of the specified tools at the tool providers' site.

This approach enables *experts* to develop complex tools in Java on the basis of a library of remotely accessible tool functionalities, as well as *newcomers* to use jETI's formal methods-based, graphical coordination environment to safely combine adequate tool functionalities into heterogeneous tools.

So far, the technology has been used to integrate jMoSel [25], a decision procedure for the verification of parametric systems, DFA-MC [18], a program analysis tool that carries out dataflow analyses of (Java) programs via model checking, and GEAR [1], our game-based model checker for modal mu-calculus.

External applications have concerned so far the integration of bioinformatic tools provided as webservices into seamless workflows [20], and the case study concerning Mediation and Choreography of the Semantic Webservice Challenge 2006 [22], [17].

VI. Conclusion

As mentioned by Hoare and Misra in [26], the time is ripe to embark on a Grand Challenge project to construct a verifying compiler. A verified software repository has been identified as an important step towards meeting such challenge. In the FCMIS view, the repository should include not only proven correct software but also tools that help establishing the correctness of the software in question. FMICS proposes to set up a collaborative demonstrator, based on the jETI technology, that shows how to provide tools for the repository and how to orchestrate different tools that were not originally designed to cooperate, among other functionalities. This endeavor would contribute both to the verifying compiler and to the uptake of formal methods technology by the industry.

Acknowledgement

We thank Ralf Nagel, Christian Kubczak, Stefan Naujokat, and Marc Njoku for their contribution to the jETI environment and to the FMICS platform.

References

[1] M. Bakera and C. Renner. jABC Model Checking Plugin. http://www.jabc.de/modelchecking/.
[2] Tim Berners-Lee, James Hendler, Ora Lassila: *The Semantic Web* Scientific American, May 2001.
[3] J.C. Bicarregui, C.A.R. Hoare, J.C.P. Woodcock. The verified software repository: a step towards the verifying compiler. Formal Aspects of Computing, 2006. To appear.
[4] The Collaborative Computational Projects (CCPs) http://www.ccp.ac.uk/
[5] The Community Z Tools Initiative czt.sourceforge.net/
[6] Eclipse Foundation: http://www.eclipse.org/
[7] Defence and Aerospace Research Partnership in High-Integrity Real-Time Systems (BAE Systems, Rolls-Royce plc, QinetiQ, and the University of York): Strand 3: Measurement and Management of HIRTS. http://www.cs.york.ac.uk/hise/darp/.
[8] GDB(TM) Human Genome Database [database online]. Baltimore (Maryland USA): Johns Hopkins University, 1990-Updated daily. Available from Internet URL http://www.gdb.org/.
[9] C. A. R. Hoare, The Verifying Compiler: a Grand Challenge for Computer Research, JACM(50) 1, pp 6369 (2003)
[10] www.omii.ac.uk/
[11] QED Pro Quo. http://www.qpq.org/.
[12] seir.sei.cmu.edu/seir/frames/frmset.help.asp
[13] SourceForge www.sourceforge.net/.
[14] Grand Challenges for Computer Research. http://www.ukcrc.org.uk/grand_challenges/index.cfm.
[15] Raja Vallée-Rai, *et al.* 1999. Soot—a Java optimization framework. *Proceedings of CASCON 1999* 125–135.
[16] UKCRC. *Dependable Systems Evolution*. http://www.fmnet.info/gc6/.
[17] C. Kubczak, R. Nagel, T. Margaria, B. Steffen: *The jABC Approach to Mediation and Choreography*, Semantic Web Services Challenge 2006, Phase I Workshop, DERI, Stanford University, Palo Alto, March 2006.
[18] A.L. Lamprecht, T. Margaria, B.Steffen: *Data-Flow Analysis as Model Checking within the jABC*, Proc. CC'06, 15th Int. Conf. on Com-piler Construction, Vienna (A), March 2006, LNCS, 3923, Springer Verlag, pp. 101-104.
[19] T. Margaria: *Web Services-Based Tool-Integration in the ETI Platform*, SoSyM, Int. Journal on Software and System Modelling, to appear, Springer Verlag (Springer Online First DOI: 10.1007/s10270-004-0072-z).
[20] T. Margaria, C. Kubczak, M. Njoku, B. Steffen: *Model-based Design of Distributed Collaborative Bioinformatics Processes in the jABC*, IEEE ICECCS 2006, Stanford, Aug. 2006, to appear.
[21] T. Margaria, R. Nagel, B. Steffen: *Remote Integration and Coordination of Verification Tools in JETI*, Proc. ECBS 2005, 12th IEEE Int. Conf. on the Engineering of Computer Based Systems, April 2005, Greenbelt (USA), IEEE Computer Soc. Press, pp. 431–436.
[22] Semantic Web Services Challenge 2006: Challenge on Automating Web Services Mediation, Choreography and Discovery - organized by DERI, Stanford (USA). http://www.sws-challenge.org/
[23] B. Steffen, T. Margaria, V. Braun: *The Electronic Tool Integration platform: concepts and design*, [24], pp. 9-30.
[24] *Special section on the Electronic Tool Integration Platform*, Int. Journal on *Software Tools for Technology Transfer*, Vol. 1, Springer Verlag, November 1997
[25] C. Topnik, E. Wilhelm, T. Margaria, B. Steffen: *jMosel: A Stand-Alone Tool and jABC Plugin for M2L(Str)*, Proc. SPIN 2006, 13th International SPIN Workshop on Model Checking of Software, Vienna (A), April 2006, LNCS 3925, Springer Verlag, pp. 293 - 298.
[26] C.A.R. Hoare, J. Misra: *Verified software: theories, tools, experiments - Vision of a Grand Challenge project*, July 2005, http://vstte.inf.ethz.ch/pdfs/vstte-hoare-misra.pdf
[27] FMICS WG homepage: http://www.inrialpes.fr/vasy/fmics/
[28] SUN Microsystems. Java WebService Developer Pack http://java.sun.com/webservices/
[29] WebServices.Org - homepage of the WebServices and SOA communities: http://www.webservices.org/
[30] W3C. SOAP http://www.w3.org/TR/SOAP/