
IST-2001-33149 (MRG)

# Final report

30th June 2005

Project start date: 1 Jan 2002
Project duration: 40 months
Project coordinator: University of Edinburgh
Project partners: University of Edinburgh, Ludwig-Maximilians-Universität München


**Funded by the European Community's "Information Society Technologies" Programme (1998–2002) under the FET proactive initiative on Global Computing.**

# Contents

# 1 Executive summary

The MRG project started on 1st January 2002 and ended on 30th April 2005. The aim of the project was to develop the infrastructure needed to endow mobile code with independently verifiable certificates describing its resource behaviour, in the form of condensed and formalised mathematical proofs of resource-related properties which are by their very nature self-evident, unforgeable, and independent of trust networks. This is the "proof-carrying code" approach to security, normally applied to enforce type safety of mobile code.

The project fully achieved its objectives by constructing a complete working infrastructure for generating and checking certificates describing the resource behaviour of programs written in a high-level programming language. This software, which is packaged into a web-based demo for easy use, is underpinned by foundations that provide a great deal of confidence in the integrity of certificates, including most crucially a complete formalized proof that the logical system used to build certificates is sound and complete.

The project partners were the University of Edinburgh (coordinator) and Ludwig-Maximilians-Universität München.

# 2 Project objectives

The ability to move code smoothly between execution sites will be a key part of the technological infrastructure of future global computing platforms. The pressure to supply and use mobile code in a global environment aggravates existing security problems and presents altogether new ones.

One particular security issue is the maintenance of bounds on quantitative resources. We believe that without technological foundations for providing such guarantees, global computing will be confined to applications where malfunction due to resource bound violation is accepted as normal and has little consequence, as with internet computation today. With more serious applications, resource awareness will become a crucial asset.

The main objective of this project was the development of the infrastructure needed to endow mobile code with independently verifiable certificates describing resource behaviour.

These certificates are condensed and formalised mathematical proofs of a resource-related property which are by their very nature self-evident and unforgeable. Arbitrarily complex methods may be used to construct these certificates, but their verification will always be a simple computation. One may compare this to the verification of alleged solutions to combinatorial problems such as Rubik's cube or the travelling salesman problem. (Note that the word "certificate" has a different connotation in computer security, relating to authentication and relying on a hierarchy of trusted secure computer systems rather than self-evident guarantees.)

Scenarios of application for the proposed framework include the following.

- A provider of distributed computational power may only be willing to offer this service upon receiving dependable guarantees about the required resource consumption.

- A user of a handheld device, wearable computer, or smart card might want to know that a downloaded application will definitely run within the limited amount of memory available.

- Third-party software updates for electronic devices such as mobile phones, household appliances, or car electronics should come with a guarantee not to set system parameters beyond manufacturer-specified safe limits.

- Requiring certificates of specified resource consumption will also help to prevent mobile agents from performing denial of service attacks using bona fide host environments as a portal.

**Objective 1** is the development of a framework in which such certificates of resource consumption make formal sense. This consists of a cost model and a program logic for an appropriate virtual machine and run time environment. This objective also includes the delineation of appropriate *resource policies*.

**Objective 2** consists of the development of a notion of formalised and checkable proofs for this logic which play the role of certificates. In particular, this involves the implementation of a proof checker.

**Objective 3** is the development of methods for machine generation of such certificates for appropriate high-level code. *Type systems* are used as the underlying formalism for this endeavour. Since resource-related properties of programs are almost always undecidable, we aim — following common practice — for conservative approximation: there will always be programs for which no certificate can be obtained although they abide by the desired resource policy.

**Objective 4** While proof-like certificates are generally desirable they may sometimes be infeasible to construct or too large to transmit. We therefore study relaxations based on several rounds of negotiation between supplier and user of code leading to higher and higher confidence that the resource policy is satisfied. This objective carries an appreciably higher risk than the others, due to the novelty of the idea; we expect, however, to obtain at least some useful results.

# 3 Methodologies

All objectives were pursued with respect to representative concrete examples of resource types. In the event we focussed primarily on heap space usage, but many results apply also to other resource types. Appropriate case studies provided timely validation of decisions made.

The project is foundational in nature; therefore the main outcome was publications in scientific fora. However, in order to enhance subsequent exploitation and to validate decisions made, all stages were implemented within a prototype system whose various components are available as free downloadable software.

# 4 Project results and achievements

The project fully achieved its objectives by constructing a complete working infrastructure for generating and checking certificates describing the resource behaviour of programs written in a high-level functional programming language. This software, which is packaged into a web-based demo for easy use, is underpinned by foundations that provide a great deal of confidence in the integrity of certificates, including most crucially a complete formalized proof that the logical system used to build certificates is sound and complete.

Some of the tasks undertaken in the course of the project proved to be far more challenging than originally anticipated, requiring considerably more manpower than planned. As a way of coping with the situation, a revised workplan was agreed at the beginning of the final year of the project. This involved eliminating some tasks that were peripheral to the core objectives of the project in order to allow more time to be spent on critical tasks, and adding some important additional tasks

that did not appear in the original workplan; it was not necessary to sacrifice or compromise on any of the original project objectives. This revised plan was successfully followed in the remainder of the project with all deliverables produced; see Section 5 below for specific technical results.

Interaction with other projects was through shared personnel, collaboration, and exchange of ideas. Such projects included the APPSEM-II thematic network, the ConCert project at Carnegie Mellon University, and Global Computing projects AGILE, MIKADO, PROFUNDIS, DEGAS, MyThS, DART and SECURE.

A number of companies have shown interest in the results of the MRG project. Now that we have shown how to apply proof-carrying code to resource bounds, follow-on projects will concentrate more on concrete applications (smart cards, mobile phones, the Grid) and bringing the results closer to market. See Section 7 below for details.

## 5 Deliverables and references

The following paper gives a complete overview of the MRG project at its midpoint:

> D. Aspinall, S. Gilmore, M. Hofmann, D. Sannella and I. Stark. Mobile resource guarantees for smart devices. *Proc. of the Intl. Workshop on Construction and Analysis of Safe, Secure, and Interoperable Smart Devices: CASSIS 2004*. Springer LNCS 3362, 1–26 (2005).

Now that the project has been completed, an updated and extended version of this will appear in due course.

### Workpackage 1: Virtual machine and cost model

**Objectives:** Definition of virtual machine platform, formalization of cost model, and collection of examples.

The overall goal of the project is to build a framework for Java-style downloadable bytecode equipped with checkable certificates regarding their usage of resources. The executable part of these is bytecode for a specific virtual machine. Claims about resource usage refer to the cost model, which specifies the amounts of various kinds of resource that are consumed during execution. For the sake of connecting with current practice, we employed a version of the Java Virtual Machine Language. We investigated the suitability of Microsoft's .NET intermediate language as an alternative platform at an early stage and found a way of achieving platform independence.

The main achievement in this workpackage was the definition and implementation of the Grail intermediate language. On one hand Grail can be seen as a more abstract version of our JVML subset with implemented translations to and from JVML. This more abstract view is more appropriate for the formulation and application of logical rules, see WP2. On the other it gives a degree of platform independence since it seems clear that translations to and from .NET or any similar bytecode language would not be difficult to provide. Grail is described in the following paper:

> L. Beringer, K. MacKenzie and I. Stark. Grail: a functional form for imperative mobile code. Proc. 2nd EATCS Workshop on Foundations of Global Computing, Eindhoven. *Electronic Notes in Theoretical Computer Science*, Vol. 85, Issue 1 (2003).

### Workpackage 2: Definition of bytecode logic

**Objectives:** Development of bytecode logic, including language of assertions and proof rules, and a proof checker.

The purpose of this workpackage is to provide a language for making assertions about the resource usage of bytecode programs with respect to the cost model of WP1, and a logic for proving such assertions. The certificates attached to dowloadable bytecode are proofs in this logic. Certificates must be easily checkable by the recipient, and the implementation of the proof checker is part of the trusted code base.

The main achievement in this workpackage was the definition and implementation of a VDM-style logic for Grail encoded in Isabelle/HOL, together with formalized proofs of soundness and (relative) completeness with respect to the operational semantics of Grail. These formalized and machine-checked proofs provide an extremely high level of confidence in the formulation of the logic. The bytecode logic is described in the following paper:

> D. Aspinall, L. Beringer, M. Hofmann, H.-W. Loidl and A. Momigliano. A program logic for resource verification. *Proc. 17th Intl. Conf. on Theorem Proving in Higher Order Logics (TPHOLs2004)*. Springer LNCS 3223, 34–49 (2004).

A journal version of this paper is nearly complete.

### Workpackage 3: Design of experimental high-level language

**Objectives:** Design and implementation of high-level programming language targeted at the bytecode language of WP1, to provide a test bed for WP4–7.

The preceding workpackages provide a grounding for resource guarantees on bytecode; but this is too low a level for practical programming. The objective of this workpackage was to write a compiler for a high-level programming language targeted at the bytecode language of WP1. This provides a test bed for the higher-level developments of later workpackages. The compiler is small enough to allow for rapid progress, yet sufficiently expressive to demonstrate that scaling up to real languages is possible.

The main achievement in this workpackage was the definition and implementation of the Camelot high-level language, including extensions to incorporate optimisations and object-oriented features. The object-oriented extension, O'Camelot, provides access to the Java libraries; this makes it possible to write programs that run on small devices, where input/output requires use of a device-specific GUI, as well as programs that use threads. Camelot and O'Camelot are described in the following papers:

> K. MacKenzie and N. Wolverson. Camelot and Grail: resource-aware functional programming on the JVM. *Trends in Functional Programing*, Intellect, Vol. 4, 29–46 (2004).
>
> N. Wolverson and K. MacKenzie. O'Camelot: Adding objects to a resource aware functional language. *Trends in Functional Programing*, Intellect, Vol. 4, 47–62 (2004).
>
> S. Gilmore. Extending Camelot with mutable state and concurrency. *Proc. 4th Intl. Conf. on Computational Science*. Springer LNCS 3038, 306–313 (2004).
>
> S. Gilmore, K. MacKenzie and N. Wolverson. Extending resource-bounded functional programming languages with mutable state and concurrency. *Parallel and Distributed Computing Practices* (2005).

### Workpackage 4: From reasoning principles to high-level type systems

**Objectives:** Develop reasoning principles and type systems for characterising resource usage, including a typechecker and soundness proofs.

This workpackage builds on the foundational strands in the first three packages. Beginning from the experimental high-level language of WP3, we investigated ways of expressing resource constraints and proving that they are satisfied by the compiled program, according to the cost model of WP1.

Our approach was to employ type systems where the type-checking problem is decidable, whereas the problem of proving that a resource constraint is satisfied will generally be undecidable. This means accepting an unavoidable gap (the "slack") between the set of programs which are typable in a resource type system and the larger set of programs which satisfy the resource property of interest. For many natural examples, though, the resource bounds are met for obvious reasons which are in the scope of our type systems. The craft of designing type systems lies in capturing these natural examples and minimising the slack, while retaining a practical notion of type and practical type-checking algorithms.

The main achievement of WP4 was the development of a number of type systems for describing heap space usage, representing various points in the tradeoff between the complexity of the type system, including the readability of the types assigned to programs, and the "slack". Some of this work is described in the following papers:

> M. Konečný. Functional in-place update with layered datatype sharing. *Proc. 6th Intl. Conf. on Typed Lambda Calculi and Applications*. Springer LNCS 2701, 195–210 (2003).
>
> R. Amadio. Max-plus quasi-interpretations. *Proc. 6th Intl. Conf. on Typed Lambda Calculi and Applications*. Springer LNCS 2701, 31–45 (2003).
>
> R. Atkey. A calculus for resource relationships. *Proc. 2nd Workshop on Semantics, Program Analysis and Computing Environments for Memory Management*, Venice (2004).

Some preliminary work on a type system for stack space usage is described in the following paper:

> B. Campbell. Folding stack memory usage prediction into heap. *Proc. Quantitative Aspects of Programming Languages Workshop* (2005).

### Workpackage 5: Further high-level and low-level type systems

**Objectives:** Generalise type systems in WP4 to accommodate more general notions of resource, and develop type systems for expressing resource bounds at the byte-code level.

This workpackage continued and expanded on the work begun in WP4. The idea here was to begin to generalise the systems for space-like resources studied there to consider more general notions of resource. As a particular case, we examined type systems for expressing limits on parameter values.

In the original workplan, this workpackage also aimed to broaden the scope of the type systems to consider *low-level* type systems for expressing resource bounds at the level of the VM byte code. This peripheral investigation was abandoned in the revised workplan.

The main achievement in WP5 was work on a type system for expressing limits on parameter values, reusing existing technology including Hongwei Xi's Dependent ML. This is described as an extended example of an automobile control system with safety constraints in the following paper:

D. Aspinall and M. Hofmann. Dependent types. Chapter 2 in *Advanced Topics in Types and Programming Languages* (B. Pierce, ed.). MIT Press, 45–86 (2005).

### Workpackage 6: Generation of certificates

**Objectives:** Define format of certificates and implement a certificate generator. Experiment with reducing size of certificates.

This package was concerned with generating guarantees of resource boundedness. The guarantee, or certificate, is what is shipped together with the code, as irrefutable evidence for the consumer that the code obeys the desired resource constraints. Here we considered the format of the certificates, and their generation. In the revised workplan, a task was added to address the fact that the bytecode logic developed in WP2 is a logic of *partial* correctness, meaning that any non-terminating program satisfies any property.

The main achievement of WP6 was the development of a method for automatically generating certificates of heap space usage from typing derivations and its implementation and integration with the Camelot compiler and the implemented Grail logic. This work is described in the following papers:

L. Beringer, M. Hofmann, A. Momigliano and O. Shkaravska. Towards certificate generation for linear heap consumption. *Proc. ICALP/LICS Workshop on Logics for Resources, Processes, and Programs (LRPP2004)*, Turku (2004).

L. Beringer, M. Hofmann, A. Momigliano and O. Shkaravska. Automatic certification of heap consumption. *Proc. 11th Intl. Conf. on Logic for Programming, Artificial Intelligence, and Reasoning, LPAR 2004*, Montevideo. Springer LNCS 3452, 347–362 (2005).

### Workpackage 7: Advances in high-level type systems

**Objectives:** Improve expressiveness, user-friendliness, and accuracy of the type systems developed in WP4 and WP5.

The goal of this workpackage was to improve expressiveness, user-friendliness and accuracy of the type systems developed under WP4 and WP5. The main topics of investigation were type inference and extensions to the basic type system to cope with language extensions introduced in WP3.

The main achievement of WP7 was the development, implementation and integration with Camelot of an automatic type inference algorithm for heap space types. The algorithm and its implementation are described in the following papers:

M. Hofmann and S. Jost. Static prediction of heap space usage for first-order functional programs. *Proc. 30th ACM Symp. on Principles of Programming Languages*. ACM Press, 185–197 (2003).

S. Jost. `lfd_infer`: an implementation of a static inference on heap space usage. *Proc. 2nd Workshop on Semantics, Program Analysis and Computing Environments for Memory Management*, Venice (2004).

### Workpackage 8: Integration with existing security model

**Objectives:** Implement resource manager and relate proof-checking infrastructure to present-day security management.

The preceding workpackages provide a proof-checking infrastructure which advances the state of the art in security management capabilities. This workpackage enriches our understanding of this infrastructure by relating it to present-day security management.

There are a number of significant achievements under WP8. Probably the most visible one was the implementation of a web-based demonstration system of the MRG proof-carrying code infrastructure that integrates most of the software produced by the project. This allows a user to generate Grail and JVML code for a Camelot program as well as to infer its heap space type and generate a certificate with a proof that its heap space consumption is as stated by its type. On the consumer side, JVML code equipped with a certificate is unpacked and the proof checked for validity, with execution permitted only if the proof is valid.

> H.-W. Loidl. MRG web demonstration. `http://lionel.tcs.ifi.lmu.de/mrg/pcc/` (2005).

Additional work was done on resource policies, on integrating the MRG certificate checker with the Java virtual machine:

> S. Gilmore and M. Prowse. Proof-carrying bytecode. *Proc. 1st Workshop on Bytecode Semantics, Verification, Analysis and Transformation (BYTECODE '05)*, Edinburgh (2005).

and on the use of Markovian analysis in estimating the impact of garbage-collection ignorant native methods in the JVM on managed object allocations:

> S. Gilmore and O. Shkaravska. Estimating the cost of native method calls for resource-bounded functional progamming languages. *Proc. Workshop on Trends in Functional Programming (TFP2004)*, Munich (2004).

### Workpackage 9: Reducing size of certificates

**Objectives:** Explore alternatives to 100%-guaranteed certificates when these are infeasible.

This package was concerned with exploring possible alternatives in situations where the generation and transmission of 100%-guaranteed certificates is infeasible for one of the following reasons:

- certificates exist, but are prohibitively large
- certificates can in principle be obtained, but only at prohibitively high cost (time and human resource needed for theorem proving, runtime of program analyses)
- certificates can in principle not be obtained due to influence of unknown or merely estimable parameters.

In the original workplan, this workpackage contained several tasks. In the revised workplan, all but one of these was abandoned in order to focus attention on other workpackages. This was partly because it was found that size of certificates did not appear to be a stumbling block for the examples under consideration, and partly because the use of so-called "oracle strings" allows for a drastic reduction in the size of proofs.

As a result, the only work done under WP9 was the formulation and implementation of a probabilistically checkable proofs (PCP) framework for certification of arbitrary type systems. This was a negative result in the sense that reductions in size brought about by this method only take effect with proof sizes well beyond what is currently required. It is interesting that this work appears to be the first actual implementation of PCP, despite the fact that the idea has attracted considerable attention over the past several years. This work will be written up for publication in the near future.

### Workpackage 10: Mobile virtual machines

**Objectives:** Investigate extension to support downloadable virtual machines.

This workpackage developed a thread of investigation into a mechanism to support mobility between computational environments. As with WP9, this investigation is visionary and speculative. Here we are concerned with a promising technology which could provide a way to enable greatly increased interoperability of mobile software. The foundational technology is the *mobile virtual machine*, a bytecode interpreter which is itself downloaded before the bytecode application which is to be interpreted by it. One use of this technology would be to allow more advanced virtual machines to be installed between high-level language programs and the JVM. Another would be to perform upgrades on pre-installed micro virtual machines.

Two of the three tasks under WP10 were completed with the third abandoned in the revised workplan in order to save effort for other tasks.

### Deliverables list

| Del. no. | Deliverable name | WP no. | Est. person-months | Del. type | Planned delivery (month) | Actual delivery (month) |
|---|---|---|---|---|---|---|
| D1a | Definition of virtual machine platform | 1 | 1.5 | report | 2/02 | 2/02 |
| D1b | Cost model | 1 | 2.5 | report | 9/02 | 9/02 |
| D1d | Comparison of JVML with .NET for project use | 1 | 7 | report | 12/02 | 12/02 |
| D1f | Representative examples for project use | 1 | 1 | report | 1/03 | 1/03 |
| D2a/b | Bytecode logic | 2 | 31.5 | report | 9/03 | 9/03 |
| D2c | Proof checker for bytecode logic | 2 | 3 | prototype | 11/03 | 11/03 |
| D2e | Theorem prover for bytecode logic | 2 | 2.5 | prototype | 11/03 | 11/03 |
| D2f | Encoding of VM semantics in theorem prover [optional task] | 2 | 0.5 | prototype | 5/03 | 5/03 |
| D3a | Definition of experimental high-level language | 3 | 4 | report | 5/02 | 5/02 |
| D3b | Compiler for high-level language | 3 | 4.5 | prototype | 1/03 | 1/03 |

| Del. no. | Deliverable name | WP no. | Est. person-months | Del. type | Planned delivery (month) | Actual delivery (month) |
|---|---|---|---|---|---|---|
| D3d | Extend compiler with immutable objects and higher-order functions | 3 | 5.8 | prototype | 11/03 | 11/03 |
| D3e | Extend compiler with optimisations | 3 | 2.5 | prototype | 9/03 | 9/03 |
| D3f | Extend with mutable state and concurrency [optional task] | 3 | 1 | prototype | 2/04 | 2/04 |
| D4a | Reasoning principles for resource usage | 4 | 4 | report | 12/02 | 12/02 |
| D4b/c | Type system for space-like resources | 4 | 9 | report | 11/03 | 11/03 |
| D4e/f | Soundness of type system | 4 | 11 | report | 4/04 | 5/04 |
| D5a/b | Type system for expressing limits on parameter values | 5 | 4 | report | 2/04 | 2/04 |
| D6a | Certificate generator | 6 | 27.5 | prototype | 5/04 | 12/04 |
| D6g | Total correctness | 6 | 18.7 | prototype | 4/05 | 3/05 |
| D7a | Extension of type system by allowing user annotations [optional task] | 7 | 4 | report | 6/04 | 1/05 |
| D7c | Extension of basic type system to object-oriented language | 7 | 6 | prototype, report | 9/04 | 6/05 |
| D7d | Methods to infer type annotations automatically | 7 | 10 | report, maybe prototype | 5/04 | 1/05 |
| D7e | Extension of basic type system to mutable state and concurrency | 7 | 2 | report | 9/04 | 3/05 |
| D8a | Relationship between proof-based certificates and present-day security management | 8 | 6 | report | 10/04 | 2/05 |
| D8b | Certificate-based resource manager | 8 | 6.5 | prototype | 4/05 | 3/05 |
| D8d | Methods for estimating cost of native methods | 8 | 2 | report | 8/04 | 1/05 |
| D8e | Expressing and relating resource policies | 8 | 4 | report | 4/05 | 5/05 |
| D9b/c | Advanced techniques of certification | 9 | 2.5 | report | 12/04 | 3/05 |
| D10a | Practical effectiveness of mobile virtual machines | 10 | 2 | report | 3/03 | 3/03 |
| D10b | Metalanguage for describing virtual machine configuration | 10 | 1.3 | report | 2/04 | 2/04 |
| D11a | Project website | 11 | 0.5 | website | 2/02 | 2/02 |
| D11b | Kickoff workshop | 11 | 0 | workshop | 5/02 | 5/02 |

| Del. no. | Deliverable name | WP no. | Est. person-months | Del. type | Planned delivery (month) | Actual delivery (month) |
|---|---|---|---|---|---|---|
| D11c | Workshop at end of year 1 | 11 | 0 | workshop | 12/02 | 2/03 |
| D11d | Workshop at end of year 2 | 11 | 0 | workshop | 12/03 | 3/04 |
| D11e | Workshop at end of year 3 | 11 | 0 | workshop, proceedings | 4/05 | 4/05 |
| D11f | Measurable criteria of progress/success | 11 | 0 | report | 6/02 | 8/02 |
| D11g | Assessment of progress in year 1 | 11 | 0 | report | 12/02 | 1/03 |
| D11h | Assessment of progress in year 2 | 11 | 0 | report | 12/03 | 2/04 |
| D11i | Final assessment of progress | 11 | 0 | report | 4/05 | 6/05 |
| D11j | Dissemination and use plan | 11 | 0 | report | 6/02 | 6/02 |
| D11k | Technological implementation plan | 11 | 0 | report | 4/05 | 6/05 |

# 6 Potential impact of project results

| Questions about project's outcomes | No. | Comments or suggestions for further investigation |
|---|---|---|
| **1. Scientific and technological achievements of the project (and why are they so)** | | |
| Question 1.1.<br>Which is the 'Breakthrough' or 'real' innovation achieved in the considered period | N/A | We have developed a complete prototype implementation of a PCC infrastructure involving a number of innovative components. Programs are written in the high-level functional Camelot language and compiled by a certifying compiler which translates the program to the Grail intermediate language and thence to bytecode. The compiler also performs heap-space usage inference by means of the Hofmann-Jost method and uses this to generate a certificate (expressed in a bytecode logic) that the compiled version of the program satisfies the given resource bound. The bytecode program and the certificate are then packaged together in a JAR file for transmission to the code consumer. The process of space inference and certificate generation is entirely automatic, requiring no human intervention. When the code consumer receives the JAR file, the bytecode program is converted back to its Grail representation and the proof is then examined in order to verify that the program does indeed satisfy the claimed resource usage. If the proof is found to be valid then the program can be executed in the normal way and the consumer can be confident that the stated resource bounds will not be exceeded. |
| **2. Impact on Science and Technology: Scientific Publications in scientific magazines** | | |
| Question 2.1.<br>Scientific or technical publications on reviewed journals and conferences | 15 | Details in Section 5 |
| Question 2.2.<br>Scientific or technical publications on non-reviewed journals and conferences | 1 | Details in Section 5 |

| Questions about project's outcomes | No. | Comments or suggestions for further investigation |
|---|---|---|
| Question 2.3.<br>Invited papers published in scientific or technical journal or conference. | 2 | Details in Section 5 |
| **3. Impact on Innovation and Micro-economy** | | |
| **A - Patents** | | |
| Question 3.1.<br>Patents filed and pending | 0 | |
| Question 3.2.<br>Patents awarded | 0 | |
| Question 3.3.<br>Patents sold | 0 | |
| **B - Start-ups** | | |
| Question 3.4.<br>Creation of start-up | No | |
| Question 3.5.<br>Creation of new department of research (ie: organisational change) | No | |
| **C - Technology transfer of project's results** | | |
| Question 3.6.<br>Collaboration/partnership with a company? | Yes | Trusted Logic, France Telecom and SAP, and possible further collaborations under discussion; details in Section 7 |
| **4. Other effects** | | |
| **A - Participation to Conferences/Symposium/Workshops or other dissemination events** | | |
| Question 4.1.<br>Active participation to Conferences in EU Member states, Candidate countries / NAS. | 16 | Details in Periodic Progress Reports |
| Question 4.2.<br>Active participation to Conferences outside the above countries | 2 | Details in Periodic Progress Reports |

| **Questions about project's outcomes** | **No.** | **Comments or suggestions for further investigation** |
|---|---|---|
| **B - Training effect** | | |
| Question 4.3.<br>Number of PhD students hired for project's completion | 3 | Field: Computer Science |
| **C - Public Visibility** | | |
| Question 4.4.<br>Media appearances and general publications (articles, press releases, etc.) | 2 | `http://istresults.cordis.lu/index.cfm/section/news/tpl/article/BrowsingType/Features/ID/76759`<br>IST Results website article<br>`http://www.evca.com/images/attachments/tmpl_27_art_33_att_802.pdf`<br>EVCA Barometer article |
| Question 4.5.<br>Web-pages created or other web-site links related to the project | 3 | `http://www.lfcs.ed.ac.uk/mrg`<br>MRG project website<br>`http://www.lfcs.ed.ac.uk/camelot`<br>Camelot compiler download site<br>`http://www.lfcs.ed.ac.uk/gc`<br>Global Computing summer school |
| Question 4.6.<br>Video produced or other dissemination material | 2 | `http://www.lfcs.ed.ac.uk/mrg/project-info/flyer-colour.pdf`<br>Publicity flyer<br>`http://www.lfcs.ed.ac.uk/mrg/project-info/NeSCposterA4.eps`<br>Poster |
| Question 4.7.<br>Key pictures of results | 0 | |
| **D - Spill-over effects** | | |
| Question 4.8.<br>Any spill-over to national programs | Yes | UK e-science programme; details in Section 7 |
| Question 4.9.<br>Any spill-over to another part of EU IST Programme | Yes | FET-Open and FP6 FET; details in Section 7 |
| Question 4.10.<br>Are other team(s) involved in the same type of research as the one in your project ? | Yes | An up-to-date list of related projects is on the project website |

# 7 Future outlook

A number of companies have shown interest in the results of the MRG project. These include: Gemplus, NTT and Trusted Logic (smart cards), Helixion (secure multimedia), Motorola (automotive networks) and AbsInt (embedded systems), among others. In the closing stages of the project, we have been attempting to cement relationships with some of these companies by setting up collaborative follow-on projects. So far, the following attempts to secure funding have been successful or seem likely to succeed:

- EmBounded is an FET-Open STREP project on resource bounds in embedded systems, building on the results of the MRG project and specifically work on Grail and the Grail bytecode logic. Partners include LMU München and AbsInt.
- MOBIUS is a FET-GC2 Integrated Project proposal that will build on the results of MRG. Partners include Edinburgh and LMU München and industrial partners include Trusted Logic, France Telecom and SAP. An industrial User Panel includes about a dozen companies from a range of relevant sectors of industry.
- ReQueST is an EPSRC-funded project at Edinburgh under the "Research in the Fundamental Computer Science for e-Science" Programme. It will apply the results of the MRG project to resource certification in the Grid.
- The Edinburgh site have a proposal for a collaborative project with Helixion (a local SME in the mobile phone sector) on digital rights management and secure multimedia.

Now that we have shown how to apply proof-carrying code to resource bounds, these follow-on projects are focusing more on concrete applications (smart cards, mobile phones, the Grid) and bringing the results closer to market. Scientific work includes moving to use Java in place of Camelot as the high-level language, consideration of other resource types and seeing how the framework can be generalised to handle different kinds of resource in a uniform way.