European Commission

Title page

INFORMATION SOCIETY TECHNOLOGIES (IST) PROGRAMME

Contract for:

Shared-cost RTD

Annex 1 - "Description of Work"

Project acronym: MRG
Project full title: Mobile Resource Guarantees
Proposal/Contract no.: IST-2001-33149
Related to other Contract no.: *(to be completed by Commission)*

Date of preparation of Annex 1: 13 September 2001

Operative commencement date of contract: *(to be completed by Commission)*

Contents

1 Project summary

Objectives

Objective 1: Development of a framework for formal certificates of resource consumption, consisting of a cost model and a program logic for an appropriate virtual machine. In the first instance this will be a subset of the Java VM; later we will consider appropriate parametrisations allowing for mobile virtual machines.
Objective 2: Development of a notion of formalised and checkable proofs for this logic which will play the role of certificates, including the implementation of a proof checker.
Objective 3: Development of methods for machine generation of certificates for appropriate high-level code, either fully automatically or based on user-supplied annotations e.g. in the form of invariants. Type systems will be used as the underlying formalism for this endeavour.
Objective 4: Study relaxations of proof-based certificates based on several rounds of negotations between supplier and user of code leading to higher and higher confidence that the resource policy is satisfied.

Description of work

This project aims at developing the infrastructure needed to endow mobile code with independently verifiable certificates describing resource behaviour. These certificates will be 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.

The workplan consists of the following central tasks:

1. Define expressive formalised resource policy (cost models).
2. Define notions of independently verifiable certificate (resource sensitive program logic with proof objects).
3. Foundations for efficient generation of certificates (type systems, identification of useful programmer annotations).
4. Foundations for alternatives to generation of full certificates (proof-theoretic compression, probabilistically checkable proofs, game-theoretic approaches).

Where appropriate, each foundational task is accompanied by a prototype implementation and case studies. Apart from this, the project includes the following separate engineering-oriented tasks:

1. Design of runtime environment including virtual machine, bytecode, implemented program logic.
2. Design and implementation of a high-level programming language in which to write resource-certified code.
3. Generation and integrated use of formalised certificates.
4. Parametrisation by arbitrary runtime environment.

The deliverables are research papers describing our solutions to foundational problems and a working prototype which will be made available as free downloadable software.

Milestones and expected results

Cost model; implemented bytecode logic; implemented experimental high-level language; implemented type system for space-like resources; soundness proofs; type system for parameter value constraints with soundness proofs; implemented resource type system for bytecode; certificate generator; experiments with reducing certificate size; implemented resource manager.

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 some 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.

This project aims at developing well-founded methods to spur technological progress in this presently under-studied area. The main objective of the project is the development of the infrastructure needed to endow mobile code with independently verifiable certificates describing resource behaviour.

These certificates will be condensed and formalised mathematical proofs of a resource-related property which are by their very nature self-evident and unforgable. 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 will consist of a cost model and a program logic for an appropriate virtual machine and run time environment. In the first instance this will be a subset of the Java virtual machine. Later on we will consider appropriate parametrisations thus allowing for arbitrary runtime environments including mobile virtual machines.

The program logic must allow the following abstract resource-related properties to be expressed: runtime, space usage, number of calls to a particular function, maximum value of particular function arguments. 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 will 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. Such generation of certificates may either be fully automatic or be based on certain user-supplied annotations e.g. in the form of invariants. *Type systems* will be used as the underlying formalism for this endeavour. This objective is the most comprehensive of all and will be allocated the most effort. 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 propose to 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.

Methodology and deliverables All these objectives will be pursued with respect to four representative concrete examples of resource policies: two of them time-like, i.e., with additive behaviour with respect to composition of program parts; and two space-like, i.e., for which the consumption of a composite program is obtained as the maximum of the resource consumptions of its individual parts. We anticipate, however, considerable synergies between these four examples as large parts of the required framework overlap. Appropriate case studies taken e.g. from the JavaCard distribution [36] and its applications, e.g. [25] and/or algorithms for scientific computation [23], will provide timely validation of decisions made.

The project is foundational in nature; therefore the main outcome will consist of publications in scientific fora of which we expect about 20 overall. However, in order to enhance subsequent exploitation and to validate decisions made, we envisage implementation of all stages within a prototype system whose various components will be made available as free downloadable software.

3 Participant list

Partic. Role	Partic. no.	Participant name	Participant short name	Country	Date enter project	Date exit project
C	1	University of Edinburgh	UEDIN	UK	Start of project	End of project
P	2	LMU München	LMUMUN	D	Start of project	End of project

4 Contribution to programme / key action objectives

This project is a contribution to the Global Computing (GC) FET Proactive Initiative 2001 (Action Line IST–2001–VI.2.2). In this section we describe how the project meets the requirements of this Action Line.

First, we directly address the scientific goals mentioned in the GC Terms of Reference. Our work on managing the resource behaviour of mobile code targets the central scientific aim, which is to:

> *develop the theoretical foundations needed to enable the design of these systems in the future and to cope with the many complex issues raised by their construction.*

The foundational contributions of our proposed work, such as resource-sensitive type systems (Objective 3), and the logical underpinnings for proof-like certificates (Objectives 2,4), address a particular such complex issue, namely managing the resource usage of mobile code and entities.

Our research tackles the three key aspects mentioned in the IST Work Programme 2001, Action Line VI.2.2:

1. *The design of systems composed of autonomous entities [...]* — we will build a framework for ensuring resource bounds on mobile code requiring no centralized control.

2. *Analysing and reasoning about the [quantitative] behaviour of such systems [...]* — quantitative analysis of the resource-related behaviour of programs forms the backbone of our proposed work.

3. *Avoiding and/or detecting undesirable behaviour [...]* — we will prohibit systems interacting when this would lead to a resource violation.

Moreover, our project directly addresses the following pervasive important issue mentioned in the Work Programme:

- *[...] controlling the use of resources [facilitating] the utilisation of the resources in a transparent way [...]* — our scheme will allow precise control over the utilisation of resources without introducing any dynamic overhead.

In considering the underlying aspects of the technological infrastructure, our work addresses a broad range of systems, rather than any specific application area. Our research will be relevant for the infrastructure underlying mobile computing devices such as telephones, the interaction involved in road and air traffic management, as well as the more general context of computational grids, where contracts and predictions of resource usage may be especially relevant.

The issue of resource awareness is also mentioned as an example research area in the final section of the GC Terms of Reference. Our work will investigate protocols for calculating resource bounds, communicating them, and ensuring that they are met. As an application of this, agents will be able to schedule their computation by dynamically yet accurately allocating resources where and when they are needed.

5 Innovation

This project is innovative both in its central concept and in the detailed techniques to be developed in support of that concept.

The overall aim of the project is to investigate a novel method for controlling resource usage by mobile computational entities. Recently, it has become common practice to accompany mobile code with cryptographic certificates that make assurances of security; for example, Java applets or Microsoft Authenticode signatures. These rely upon pre-existing networks of trust, and are usually concerned with assuring the user that code comes from a particular author. By contrast we propose to send certificates that demonstrate concrete properties of the code itself; recipients can check these directly, without the need for any trusted authority.

The novelty lies in this independence from trust, and particularly in the focus on managing resources. We may trust a code supplier not to write malicious code, but need an additional local check that it will not exceed the resources available. Thus our approach does not replace existing practices, but complements and extends them. In some cases this can add a layer of certainty previously unobtainable: for example, we hope to guarantee that some kinds of calculation will produce their answer within a given time; all a runtime check can do is to turn off the computation after a time limit.

The project involves several interacting workpackages, each contributing a part towards this goal. Several of these comprise innovative work, which we look at now. The workpackages are described in detail in Section 9.3 below.

- Packages 1 and 2 will develop a cost model and accompanying logic for a virtual machine. The increasing use of such virtual machines, and their high-level open specifications, means that we can attempt a *portable* resource analysis, previously infeasible. This promises significant returns in expanding the application of such work; it carries a certain risk that some VM implementations may not properly match their specification.

  Focusing on the bytecode level also allows a degree of *language independence*, so that in the long term our work can be applied to more than one programming paradigm, mapping to a common cost model.

- Package 3 lays the foundation for translating resource information between low-level bytecode and high-level languages. This is through a new transformational compiler that in Package 6 will be extended to carry resource information along with code as it is compiled.

- The type systems for resource analysis to be developed in packages 4, 5 and 7 aim for a precision and expressiveness ahead of current work. We also plan to look at a broader notion of resource than addressed by existing systems: use of file handles, network resources, number of system calls, value of system call parameters.

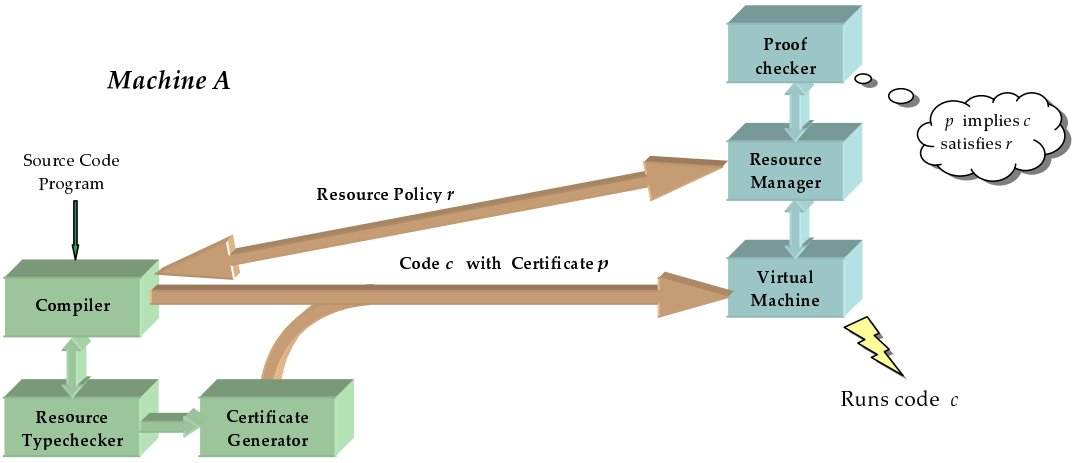
- Packages 6 and 9 involve investigating a range of novel techniques for managing the proofs delivered in our certificates. The aim here is to reduce the load associated with sending certificates, either by reducing the size of proofs themselves, or through protocols that can avoid transmitting entire proofs. Individual methods carry the risk that they may turn out to be impractical, but each has the possibility of significant long-term return.

- Active security management and security policies are increasingly important in current systems for code distribution. Package 8 investigates novel extensions of this to *resource* management and policies, with the significant feature that our certificates can guarantee that run-time checks are not required.
- Portable guarantees depend on well-understood virtual machines, and package 10 takes this to a new level by investigating the approach of making the VM itself customisable, downloaded and configured over the network. This is looking a long way forward, with corresponding risks, but the possibility of considerable returns in matching desired performance to the available resources.

5.1 A novel protocol for resource certification

We propose a novel protocol for certifying the resource bounds of mobile code. It is designed so that it can be integrated with the existing way that a Java virtual machine ensures that security constraints are met, via the *Security Manager*. In Java, the Security Manager is responsible for enforcing the security policy requested by the user, by checking that no violations of security are made while the code runs.

In our protocol, a *Resource Manager* is responsible for verifying that the certificate supplied with a piece of code ensures that it will run within the constraints required. A *Proof Checker* is invoked to do this. If the check succeeds, we have an absolute guarantee that resource bounds are met, so it is not necessary to check for resource violations as the code runs. Here is a diagram representing the components which feature in our protocol, assuming a source machine ***A*** where the code originates, and a machine ***B*** where the code will be run:



To allow extra flexibility in the framework, we also suppose that the resource policy can be negotiated with the code producer. In detail, the phases in the protocol are:

- **Initiate:** *A* wants to send *B* a piece of mobile code *c*.
- **Policy:** *A* and *B* agree upon some *resource policy r* that the code must satisfy. The choice of policy may influence some aspect of the compilation (or re-compilation) of the code from a high-level language, in particular how the Resource Type-checker influences the Certificate Generator.

- **Certify:** Provided the code meets policy r, then A sends B the code c together with a *certificate* p that c abides by r.
- **Check:** B checks the validity of the certificate p with respect to the code and the agreed resource policy.
- **Run:** Provided the check was successful, B then runs the code.

Initiate may mean that A and B know each other and enter into email contact; it may mean that A publicly advertises her code for download or it may mean that either A or B or both are in fact non-human agents which act according to some predefined protocol.

Policy may mean that A requests a certain resource policy based e.g. on her machine configuration or current load or on a more general policy prescribed by the manufacturer or system administrator. It may alternatively mean that A publicly advertises compliance with certain predefined resource policies, so no negotiation is required.

Certify will always mean that A constructs a certificate which fits a certain predefined and generally agreed format. This may be done using a combination of automatic inference and programmer annotation.

Check means that B verifies the correctness of the certificate. The generally agreed format for certificates must be such that checking is computationally easy.

Run may mean that B manually installs the code c for execution; it may also mean that the system configuration only allows code with a valid certificate to be run.

5.2 Advances beyond present state-of-the-art

This project draws together a number of threads in existing work, and looks to the future, building upon them in the novel context of resource management for autonomous mobile entities. We outline the state of the art in some relevant areas.

The general idea of packaging a piece of code with a formal proof of some of its properties is not new; it probably first arose within the EC-funded "Logical Frameworks" Basic Research Action (1989–1991) in the form of Burstall-McKinna's *Deliverables* [35]. It was, however, then ahead of its time since programs were largely monolithic and tied to a particular application site. The concept received a new boost through the work of G. Necula [42] who coined the catchy phrase *Proof-Carrying Code*, harnessed it for certified assembly code and OS applications, and for the first time demonstrated the technical feasibility of the idea (see also recently approved projects at Berkeley [15, 43], Princeton [1, 2] and Stevens Institute [16, 5].

What is new in the present project is, apart from the ambition to explicitly target mobile code, the quantitative aspect of the asserted properties. While the existing Proof-Carrying Code (PCC) projects concentrate on security and safety — making sure that mobile code does not do "bad" things — our project focuses attention on the issue of resource use, which is in many ways a separate concern: well-intentioned and safe code may still consume more resources than a user (or device) may be prepared to offer. We expect to make useful links with the ongoing PCC work, and we believe that that our distinct perspective will advance this general area of research. In particular, we expect direct collaboration with Stevens Institute, which also (uniquely until now) touches on the idea of extending PCC to enforcing resource bounds. (For comparison, we should mention that the Stevens Institute project consists of several related ideas and is a smaller-scale project than the one we propose).

We will be able to draw on some of the proof-theoretic infrastructure (to be) developed in the abovementioned PCC projects; the technical questions concerning the generation of certificates via type systems and annotations are, however, disjoint.

We plan to draw considerably on current foundational research in the general area of *resource-bounded computation*. The relevant results in this field are type-theoretic characterisations of low time and space complexity classes [8, 7, 30, 29]; static determination of time and space usage of programs typable in such formalisms [18, 32, 34, 11], and first steps towards taking space-efficient optimisations into account [34, 40, 31]. This is a currently active research area in which some of the project team are already involved, and where we aim to contribute further advances.

Some existing work on formalising virtual machines, and the JVM in particular [51, 14, 44, 19] will be useful to us. This is mostly concerned with type-safety, and in particular the role of the Java bytecode verifier. The present project goes into new territory by considering resources and associated cost models.

Our approach of using a transformational compiler, carrying resource information from program through to bytecode, builds on current work on systems that maintain type information throughout compilation [53, 38, 10] – sometimes termed *Type-Directed Compilation*. This in turn links to the concept of *Typed Assembly Language* [39, 17], where types carry information about low-level assembly code. As with the formalisation of virtual machines, most existing work addresses type-safety rather than resource issues.

A less-closely related area of work is the concept of programming for *Active Networks*, studied by the PLAN group at Penn [33]. Packets that travel over an active network carry out computation as they go; programs in PLAN are executed within the transport fabric itself. Guaranteeing tight use of resources is vital to maintain network availability, but PLAN works by constraining the high-level programming language used, rather than generating proved certificates as we do.

6 Community added value and contribution to EC policies

6.1 European dimension

One of the issues driving the proactive initiative on Global Computing is the ubiquity and sheer scale of the infrastructure that must be developed to support truly dynamic and mobile computation. The problems that arise, and the solutions to be found, will inevitably reach beyond national considerations.

The values and benefits anticipated in the vision of mobile computational entities can only be realised if the technology and standards used have sufficiently wide scope. A collection of distinct islands separated by incompatible protocols will hinder this: it is essential that research in this area involve international collaboration, and be disseminated to an international audience.

6.2 European added value

The participants at the project sites in Germany and the UK provide complementary expertise in relevant areas: neither could successfully complete the project alone. In particular Hofmann in Munich is doing essential enabling work on type systems for bounded computational complexity, some in collaboration with Aspinall in Edinburgh.

As well as the specific consortium membership, there is a wider European involvement in this research area, and in the existing work that we plan to use. Previous EC-funded working groups like TYPES, CLICS-I and II, and APPSEM provide much of the ground material that makes a project like this possible. These working groups have been extremely successful in energising a Europe-wide research community in relevant areas, which we can look forward to interacting with, drawing from, and contributing to.

6.3 Contribution to EU policies

The IST Work Programme names two specific priorities in support of EU policies on employment, social cohesion and competitiveness:

- *Fostering the convergence of information processing, communications and media*;
- *ensuring interoperability and coherence at a global level*.

This project makes contributions to each of these. Convergence of information technologies requires a common framework, which can only be developed with a foundational understanding of how different resource requirements can be supported in a dynamic environment. Equally, in a setting where code and other "active" content may move between platforms, coherence can only be achieved where there are standards for communicating reliable guarantees of resource usage. We expect that the work in this project will contribute knowledge that helps to make this possible.

7 Contribution to Community social objectives

The founding motivation for the Global Computing Proactive Initiative is the wish to support an information infrastructure which can provide continued access to continuously-updated information sources for automonous mobile entities. Although it is widely anticipated that the technological support for such systems (memory availability, bandwidth, processing power) will continue to develop at an impressively rapid pace, such systems will fail to achieve their potential unless the reliability and dependability of the applications which run on these systems develops apace with them.

Our project will contribute to building public confidence in software-reconfigurable mobile devices by providing solid foundations for a system which will guarantee that applications quantify the amount of computational resources that they will consume. Without such foundations the next generation of applications which run distributed over a hardware platform of mobile devices (which, by their nature, have severely limited computational power) will be prone to the same problems as analogous present-day systems. That is, failure due to violation of system resource limits is regarded as part of the normal operation of the system and it is widely understood that systems which are built to this design cannot be relied upon where the consequences of failure could be grave.

In this regard the present project directly addresses one of the priorities for WP2001 (dependability of systems):

> *To emphasize trust and security, including information security, privacy, suppliers and users rights and dependability of systems and infrastructures, as a general requirement for all technologies, applications and services.*

7.1 Improving the quality of life

Dependable mobile computing systems have applications in many aspects of both public services and private sector commercial activities. To name a few areas of application one could consider transport, healthcare and law enforcement. Improvements in the efficiency and effectiveness of these services bring benefits through improving the quality of life for all members of society, extending far beyond the group whose profession is primarily centred on the use of information technology.

One of the practical benefits which could flow from a foundational study such as ours is support for secure bespoke software applications which run on next-generation handheld computers such as pagers and PDAs equipped with Java virtual machines which are compactly engineered to operate in consumer electronic devices (CVMs). Such handheld devices will provide low-cost computing platforms. Their adoption of the Java technology for embedded applications already strongly supports well-managed, predicable custom software development. Our project will develop techniques and tools which would make those applications predictable and dependable enough (with respect to resource use) for their use in critical systems.

7.2 Health and safety

The most direct area of application for critical-system mobile computing is in healthcare, such as in the setting of an accident and emergency department of a hospital. Providing the attendant clinicians with access to continuously-updated multimedia data from an incoming

ambulance would be a marked advance over the present operational mode where communication with the ambulance is typically limited to voice-only. Putting the capability in the clinician's hands to control micro digital video cameras on the ambulance while simultaneously receiving streaming input data monitoring the patient's vital signs would empower them to make preliminary diagnoses and allow them to begin advance preparations for the likely programme of treatment. In such a setting any preparatory work can be life-saving.

Only a few years ago such a system would have been impractical because of the high cost of the hardware components which would be deployed. Now those costs have fallen to the point where the primary impediment to the development of such advanced healthcare systems is the difficulty of guaranteeing the dependability of the necessary control software.

The Division of Informatics in the University of Edinburgh has a particularly strong involvement in "e-health". In March 2001 it hosted a national workshop on Healthcare Informatics. This subject is one theme of the high-profile "Dependability of Computer-Based Systems" Interdisciplinary Research Collaboration in which Edinburgh is a partner and in which Sannella has some involvement. The University of Edinburgh runs a Medical Informatics programme. The Division has close links with Voxar (`www.voxar.com`), a major medical applications software provider.

7.3 Improving employment

The planned programme of work contains many opportunities for commercial exploitation of developments of the systems which it will create. Both of the institutions involved have active support infrastructure for developing technology transfer and entrepreneurial exploitation. We predict that this will be the likely route for exploitation of the ideas and new employment opportunities. Our programme of work has opportunities for students at each of the institutions involved to participate in the project and this would be a potential source of capable staff.

8 Economic development and scientific and technological prospects

The ultimate application of this work is in the development of future global computing platforms, where code is required to move across a variety of distributed devices. As stated in the call for the proactive initiative:

> *In future most objects that we work with will be equipped with processors and embedded software to perform and control a multitude of tasks in our everyday environment. . . . The envisioned systems are highly dynamic: physical devices are mobile, connectivity and bandwidth are changing, computational processes and data can migrate, and applications come and go. The availability and responsiveness of the resources that are active in an application at any given point in time are unpredictable and difficult to control.*

Even with computation in every device, connections to the network may intermittent, and there will inevitably be constraints on computing power. Exploiting these systems requires foundational understanding of how to manage the interaction between resources and mobile code.

8.1 Applications

The usefulness and potential range of applications of the research planned in this project is best illustrated by the examples suggested at the beginning of Part B:

- 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.

Systems like this will certainly be built: what this project aims to provide is a part of the theoretical infrastructure that can make them reliable and successful.

8.2 Strategic impact

Current European strength in mobile and wireless technologies means that global computing applications like those above are an important potential market, of real importance for future competitiveness. This project will contribute new understanding of the principles and limitations behind such applications, vital for their eventual development into working products and services.

The direct application of the individual workpackages is to generate results and knowledge that we shall pass through to the future projects that will build systems like these on a large scale. Several of the detailed avenues of research planned are speculative and even risky, in the anticipation that even if some are found impractical, this "negative" information is also useful to those trying to build concrete systems. As such, the strategic impact of this project will be on other, forward-looking, research projects that can draw on our work to more effectively build working systems for global computing.

8.3 Dissemination strategies

The prime audience for this work is other researchers, and consequently the main route for dissemination will be technical reports, journal papers and conference presentations by the participants. This enables rapid and timely publicity among those who can draw most from the work. We do not rule out short-term impact on programming language design in industry, but recognise from experience that this is often dependent on non-technical factors outside our control.

As well as academic papers, we plan to produce an incremental series of prototype and proof-of-concept implementations, as listed in the workplan: the proof checker and theorem prover of 2c and 2e; the compiler of package 3; the type checker of 4c; the certificate generator of 6; and the integration work of package 8. Prototypes like this are an effective way to demonstrate theoretical results in a practical context, and we shall make them available (with source code) over the web.

To further promote research in this area, we shall arrange at least one workshop on mobile resource guarantees, associated with a suitable international conference.

9 Project workplan

9.1 General description

The workplan consists of the following central tasks:

1. Define expressive formalised resource policy (cost models): workpackage 1.
2. Define notion of independently verifiable certificate (resource sensitive program logic with proof objects): workpackage 2
3. Foundations for efficient generation of certificates (type systems, identification of useful programmer annotations): workpackages 4, 5, 7
4. Foundations for alternatives to generation of full certificates (proof-theoretic compression, probabilistically checkable proofs, game-theoretic approaches): workpackage 9

Where appropriate, each foundational task will be accompanied by a prototype implementation and *case studies* to be taken for instance from the JavaCard distribution [36] or the LEDA project on certified algorithms [23], see task 1f.

Apart from this we can identify the following separate engineering-oriented tasks:

1. Design of runtime environment including virtual machine, bytecode, implemented program logic: workpackages 1, 2
2. Design and implementation of a high-level programming language in which to write resource-certified code: workpackage 3
3. Generation and integrated use of formalised certificates: workpackages 6, 8
4. Parametrisation by arbitrary runtime environment: workpackage 10

The deliverables we aim for are research papers describing our solutions to foundational problems and a working prototype implementing the above described protocol which will be made available as free downloadable software.

9.2 Workpackage list

Work-package No	Workpackage title	Lead con-tractor	Person-months	Start month	End month	Phase	Deliv-erable No
WP1	Virtual machine and cost model	1	13	0	3		D1a, D1b, D1d, D1f
WP2	Definition of bytecode logic	1	13	3	20		D2a, D2b, D2c, D2e, D2f
WP3	Design of experimental high-level language	1	13	0	27		D3a, D3b, D3d, D3e, D3f
WP4	From reasoning principles to high-level type systems	1	24	4	23		D4a, D4b, D4c, D4e, D4f
WP5	Further high-level and low-level type systems	1	18	7	33		D5a, D5b, D5d, D5e
WP6	Generation of certificates	1	17	14	25		D6a, D6c, D6d, D6f
WP7	Advances in high-level type systems	2	21	10	35		D7a, D7b, D7c, D7d, D7e
WP8	Integration with existing security model	1	8	27	35		D8a, D8b, D8d
WP9	Reducing size of certificates; negotiation vs. proof	2	18	25	35		D9a, D9b, D9c
WP10	Mobile virtual machines	1	16	4	34		D10a, D10b, D10c
WP11	Project management, dissemination and evaluation	1	1	0	36		D11a, D11b, D11c, D11d, D11e, D11f, D11g, D11h, D11i, D11j, D11k
	TOTAL		162				

The contribution of unpaid man-months per workpackage per participant is as follows.

Participant 1: WP1 1, WP2 0.5, WP3 1, WP4 0.5, WP5 1, WP6 1.5, WP7 0, WP8 0.5, WP9 1, WP10 1.5, WP11 1.5.
Participant 2: WP1 0, WP2 0.5, WP3 0, WP4 1, WP5 0.5, WP6 0, WP7 1, WP8 0, WP9 0.5, WP10 0, WP11 0.

9.3 Workpackage descriptions

This section contains a description of each workpackage. The person-months given refer to the researchers that are employed by the project; additional labour will be provided by the main investigators themselves.

Workpackage description

Workpackage number: WP1 — Virtual machine and cost model
Start date: month 0
Person-months for each participant: 9 pm for participant 1, 4 pm for participant 2

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

Description of work

This workpackage provides the basis for the research in the remaining packages. 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 will be bytecode for a specific virtual machine, to be defined in 1a. Claims about resource usage will refer to the cost model defined in 1b, which specifies the amounts of various kinds of resource that are consumed during execution. For the sake of connecting with current practice, we will employ a version of the Java Virtual Machine Language. We plan to investigate the suitability of Microsoft's .NET intermediate language as an alternative platform at an early stage (1d). If JVML turns out to be too restrictive for our purposes then we could switch attention to .NET.

a. Define **virtual machine platform**. *(2 months)*
Deliverables: internal technical document
We will define a subset of JVML that is expressive enough to permit straightforward compilation of the high-level language to be defined in 3, yet small enough that to be tractable as an object of study, admitting proofs like those to be done in 4. Some modifications or simplifying assumptions may be required.

b. Formalise **cost model**. *(6 months)*
Deliverables: technical report *Prerequisites: 1a*
Here we will define the amount of resource consumed by running a unit of mobile code. This will take the form of a semantics for our VM that delivers a cost along with the computed result, using previous work on formalisation of JVM (e.g. [51], [14], [44] and/or [19]) as a basis. Ultimately, many different kinds of resource are of interest; we hope that by considering two different concrete examples — heap space and number of system calls — the elements of a general account will emerge. There may well be a division into time-like measures that are additive with respect to independent components versus space-like measures where the cost of performing two independent computations is the maximum of the costs of the components. In practice, costs depend on the implementation of the JVM with techniques such as just-in-time optimization potentially leading to dramatic improvements. Initially we will assume that no optimizations are done, but see 7b. Included in the deliverable will be a number of illustrative examples.

c. **Milestone: Completion of cost model.** *Prerequisites: 1b*

d. Investigate **.NET as an alternative to JVML**. *(3 months)*
Deliverables: internal technical document *Prerequisites: 1b*
Microsoft's .NET platform provides a virtual machine and runtime environment that offers an alterative to JVM for supporting Java and other languages [24]. With the support of HP and Intel the .NET intermediate language is now being taken through the ECMA standardisation process. A benefit of JVML is that at present implementations are widely available, but it is designed specifically to support Java, and its built-in support for Java's type system may be a barrier to progress. Then .NET, which is designed to support a wide variety of languages and so offers extra flexibility, may turn out be more appropriate for our purposes. Here we plan to compare the suitability of .NET with JVML for the project.

e. **Milestone: Decision to use JVML** as planned **or switch to .NET.** *Prerequisites: 1d*

f. Collect some **representative examples**. *(2 months)*
Deliverables: technical report
Here we will compile a set of examples to use as test cases throughout later stages of the project. A good source of interesting algorithms and data structures is the LEDA project [23]. The amount of memory available on a smart card is severely restricted, making this an area where resource guarantees like those we aim to supply will be of interest. This means that JavaCard [36] is another source of interesting examples, based, for instance, on the e-cash examples from the JavaCard documentation or medical information systems such as [25].

The research in this package provides the first part of the infrastructure on which later packages will build.

Deliverables
D1a, D1b, D1d, D1f

Milestones and expected result
Completion of cost model
Decision to use JVML or switch to .NET (expecting to use JVML)

Workpackage description

Workpackage number: WP2 — Definition of bytecode logic Start date: month 3 Person-months for each participant: 7 pm for participant 1, 6 pm for participant 2

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

Description of work

The purpose of this workpackage is to provide a language (2a) for making assertions about the resource usage of bytecode programs with respect to the cost model defined in 1b, and a logic (2b) for proving such assertions. The certificates that will be attached to dowloadable bytecode will be proofs in this logic (but see 9). Certificates must be easily checkable by the recipient, and the implementation of the proof checker (2c) will be part of the trusted code base (TCB); thus the logic and its implementation must be simple and generally acceptable. To ensure the quality of the logic and its implementation, we have included related tasks (2e and 2f) whose main underlying aim is assessment of these components, and do not directly feed into later work.

a. Develop **language of assertions**. *(2 months)*
Deliverables: technical report *Prerequisites: 1be*
We will develop a logical language for asserting resource-related properties of bytecode programs. This will probably take the form of a Hoare-style logic, with assertions making explicit reference to code fragments. The expressiveness and convenience of the notation will be checked using examples from 1b and its semantics will be formally defined.

b. Develop **proof rules**. *(3 months)*
Deliverables: conference/journal paper *Prerequisites: 2a*
We will develop rules for proving assertions expressed in the language of 2a. The goal is rules that are simple, without computationally intractable conditions, that enable a large class of interesting assertions to be proved in a relatively straightforward way. We aim to generate proofs automatically via the use of novel type systems (see 4) so these rules will not normally be used directly by users (although see 7a). Understandability is nevertheless important to instill confidence in the integrity of our framework, and the length of proofs is also an important issue in practice (see 9). It will be essential to demonstrate that the rules are sound with respect to the cost model in 1b, and we will also attempt to check relative completeness of certain fragments.

c. Implement a **proof checker**. *(4 months)*
Deliverables: prototype implementation *Prerequisites: 2b*
Given an ostensible proof built using the rules in 2b, we need to be able to efficiently and automatically check that it is valid. For this purpose, Necula [41] uses Edinburgh

LF [26], a generic type-theoretic framework for encoding and checking proofs. Depending on how complicated the logic is, we will take the same approach or alternatively handcraft a proof checker.

Later work (for example 6d) requires that a rudimentary metalanguage be provided, so that lemmas can be proved and then referred to, or for construction of proofs by primitive recursion. Basic facilities of this kind will be provided. The estimate of work required is under the assumption that this will be relatively straightforward. Some care is required to prevent denial of service attacks based on the creation of huge proofs.

d. **Milestone: Completion of implemented logic.** *Prerequisites: 2c*

e. Implement a **theorem prover**. *(3 months)*
Deliverables: experimental implementation; case studies *Prerequisites: 2b*
Here we plan to experiment with the use of an interactive theorem prover such as Isabelle [48] for generating proofs. This will give access to the features provided by such provers, e.g. context management, goal-directed proof search, and built-in decision procedures. The main point of this work is to explore some of the ramifications of the choice of proof rules in 2b; in practice proofs will be generated by other means, see 4.

f. Optional: **encode VM semantics** in theorem prover. *(1 month)*
Deliverables: experimental implementation *Prerequisites: 2b*
Here we would encode the entire virtual machine with its semantics in a general-purpose theorem prover such as PVS [45], and then use this to *formally* establish that the proof rules in 2b are sound. The aim would be to explore some of the consequences of our earlier choices in order to validate these decisions. This could be carried out as a final year student project under the supervision of project personnel; this supervision is what is allowed for in the manpower estimate.

The research in this package is a mixture of design, assessment, and implementation work. A working proof checker (2c) will be an essential component part of the final overall system, and the quality of the logic that it implements is vital to the success of the entire framework.

Deliverables
D2a, D2b, D2c, D2e, D2f

Milestones and expected result
Completion of implemented logic

Workpackage description

Workpackage number: WP3 — Design of experimental high-level language
Start date: month 0
Person-months for each participant: 9 pm for participant 1, 4 pm for participant 2

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

Description of work

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 is to write a compiler for a high-level programming language targeted at the bytecode language of 1a. This will provide a test bed for the higher-level developments of packages 4, 5, 6 and 7. The compiler should be small enough to allow for rapid progress, yet sufficiently expressive to demonstrate that scaling up to real languages is possible.

The most distinctive requirement on the compiler is *transparency*: it must be possible to calculate how a particular fragment of high-level code will transform into bytecode, and what its resulting resource usage will be according to the cost model of 1b. This is vital to support the language-level approach of following workpackages.

As this compiler is to provide a framework for later development, it should also be *extensible*. Other packages aim to add functionality, like resource types (4c) and generation of certificates (6a). This must be possible without disturbing the basic action of the compiler. To meet these requirements the compiler needs an open and well-documented architecture, in sufficient detail to support reasoning (formal and informal) about its behaviour (4e). Consequently, the technical documentation produced by this workpackage is of particular importance, providing reference material for later work.

a. Define a base **programming language**. *(1 month)*
Deliverables: internal technical document *Prerequisites: 1a*
The language should be strongly typed, providing at least functions and recursive datatypes. It needs an accompanying operational semantics, and information about the intended mapping to bytecodes.

b. Write a **compiler** for the base language. *(4 months)*
Deliverables: prototype implementation; technical documentation *Prerequisites: 3a,1b*
The important issue here is not to produce a high-grade optimising compiler, but rather one that is predictable, clearly written, and well-documented, to support the work to follow. The documentation should include discussion of the anticipated resource usage of bytecodes generated by the compiler.

c. **Milestone: Completion of implemented programming language.** *Prerequisites: 3b*

d. Extend with **immutable objects** and **higher-order functions**. *(4 months)*
Deliverables: prototype implementation; technical documentation *Prerequisites: 3b*
These additional features make the language more expressive, and will assist more

advanced parts of later packages, like 7c. Like the base language and compiler, these extensions need comprehensive documentation: covering both the semantics of the language features themselves, and their compilation to bytecode. This particularly important for higher-order functions, as they typically fit less well with existing object-style VM's.

e. Implement well-understood **optimisations** in the compiler. *(3 months)*
Deliverables: prototype implementation; technical report *Prerequisites: 3b*
It is sensible to extend the compiler with optimisations whose effect can be determined statically. Example are some uses of in-place update, or the replacement of tail recursion by iteration. Exactly what is possible will depend on the chosen bytecode and its cost model — because of the need to keep track of resource usage in a provable way, this is a tighter constraint than for most compilers.

f. Optional: Incorporate **mutable state** and **concurrency**. *(1 month)*
Deliverables: prototype implementation; technical documentation *Prerequisites: 3d*
Although existing virtual machines provide good support for both of these features, they are less amenable to the kind of formal reasoning presented here. Concurrency in particular has a complex interaction with resource usage. This optional component is closely tied to 7e. This could be carried out as a final year student project, and the manpower estimate is for supervision time.

Deliverables
D3a, D3b, D3d, D3e, D3f

Milestones and expected result
Completion of implemented programming language

Workpackage description

Workpackage number: WP4 — From reasoning principles to high-level type systems Start date: month 4 Person-months for each participant: 8 pm for participant 1, 16 pm for participant 2

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

Description of work

This workpackage builds on the foundational strands in the first three packages. Beginning from the experimental high-level language designed in 3a, we investigate ways of expressing resource constraints and proving that they are satisfied by the compiled program, according to the cost model of 1b. We begin from reasoning principles, perhaps related to the bytecode logic, and then move towards type systems. The notion of type is a very broad one: type-checking can be used to enforce simple consistency checks (that the addition operation $+$ is always applied to two numeric arguments) but also rich semantic notions (such as interference [50], presence of side-effects [52], and resource usage as proposed here or studied in [31]).

Our approach is to stick with 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 that we accept 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.

a. Investigate **reasoning principles for resource usage**. *(6 months)*
Deliverables: technical report; case studies *Prerequisites: 1b,3a,3b*
Here we will investigate the resource behaviour of the constructs in our high-level language from 3. This involves program analysis: we will identify natural examples of programs which meet different kinds of resource bounds, and try to establish this formally by explaining how the compiler from 3b compiles these examples to byte code, and proving how the resulting byte code meets the resource bounds, using the bytecode logic. In effect, we are looking for high-level derived rules and reasoning principles for the bytecode logic, which apply directly to the high-level language. The results of this work will suggest patterns and constructs that should be built into the type systems.

b. Develop a **type system for space-like resources**. *(4 months)*
Deliverables: working design document *Prerequisites: 4a*
The first type system will be one for ensuring space-like resource bounds. There is some relevant recent research here, including [31] which describes a type system with a special *resource type* which corresponds to a unit of reusable space. Hughes and

Pareto [32] describe a type system for programming in bounded space. Crary and Weirich [18] have a type system which provides explicit bounds for time usage: the run time of a function can be expressed as a function of the input. We want to extend and adapt these systems, in particular investigating ways of attaching explicit bounds on space usage. Other resources, such as file handles, network connections, or hardware resources, could be treated in a similar way to space usage in these systems; this will be a novel approach and should help to eliminate a common class of program failures.

c. Implement **a typechecker** for the compiler described in 3b. *(4 months)*
Deliverables: prototype implementation *Prerequisites: 3b,4b*
To test our type systems on real examples, we must implement a typechecker for our high-level language. In a production compiler, implementing a typechecker can be a considerable task, especially if there is significant inference involved in type-checking. At the beginning, we will want to separate most of the issues concerning type inference, perhaps requiring extra type annotations in the source language. This means that our type-checking algorithm will be straightforward to implement, especially if we have a *syntax-directed* system where the choice of typing rule is uniquely determined by the syntax of the term we wish to type-check. We will defer more sophisticated type inference until 7d.

d. **Milestone: Implemented type system for space-like resources.** *Prerequisites: 4c*

e. Prove **soundness over the cost model**. *(6 months)*
Deliverables: technical report *Prerequisites: 4b*
It is natural to ask that our type systems should bear a close relationship to the cost model, so that they can be understood intuitively. Soundness is fundamental: a typing assertion should imply the intended cost constraints, as expressed by our model. Since the cost model applies to the byte code, we must reason about the translations made by the Compiler implemented in 3b. The task here is to conduct a detailed pencil and paper proof to validate this claim.

f. Prove **soundness over the bytecode logic**. *(4 months)*
Deliverables: technical report; conference/journal paper *Prerequisites: 4b*
By the previous part we have a direct connection with the cost model, we know that our type system can also be sound with respect to the bytecode logic for the compiled byte code. In other words, a typing assertion should imply a corresponding proposition in the bytecode logic. The job here is to prove that. It is also something of a test for the bytecode logic, since that is expressed in rather different terms than the type system. Therefore insights from the type system development may lead to tweaks in the proof rules for the bytecode logic, although in general the latter should be stronger (i.e. capable of proving more).

g. **Milestone: Soundness proofs.** *Prerequisites: 4e,4f*

The research in this package is a combination of improving existing work and combining several different ideas, previously treated in separation and with different motivations. Drawing together these strands is a vital step towards our vision.

Deliverables
D4a, D4b, D4c, D4e, D4f

Milestones and expected result
Implemented type system for space-like resources Soundness proofs

Workpackage description

Workpackage number: WP5 — Further high-level and low-level type systems Start date: month 7 Person-months for each participant: 9 pm for participant 1, 9 pm for participant 2

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.

Description of work

This workpackage continues and expands on the work begun in 4. The idea here is to begin to generalise the systems for space-like resources studied there to consider more general notions of resource. As a particular case, we will examine type systems for expressing limits on parameter values (5a).

This package also broadens the scope of the type systems to consider *low-level* type systems for expressing resource bounds at the level of the VM byte code. Although our main interest is in ensuring that resource bounds are met for programming in high-level languages, a way to help ensure this is to push the resource type information as far down as we can, and annotate the bytecode with additional typing information.

a. Develop a type system for expressing **limits on parameter values**. *(4 months)*
Deliverables: technical report; conference paper *Prerequisites: 4a*
As one of the potentially most useful concepts in our generalised concept of resource bounds, we want to have type systems which express restrictions on parameter values for functions. This generalises the most common case needed in programming: that of ensuring that an array access does not violate the bounds of the array. One particularly general way of achieving this is with so-called *dependent types* where the type expression can contain ordinary terms from the language. In practice, full dependent types lead to complex type systems which are almost always undecidable, but restricted forms of dependent types may be useful here. There is interesting related recent research [55, 6] which may help here. Apart from dependent types, there are other novel type systems for programming languages which propose ideas we might adapt, such as the notion of *shape* [9].

b. **Extend soundness proofs** for parameter value constraints *(3 months)*
Deliverables: technical report *Prerequisites: 5a,4e,4f*
The task here is to replay the proofs from 4e and 4f for the new kinds of resource type. This will exercise the generality of our framework; we hope that the earlier proofs will already be amenable to this kind of generalization.

c. **Milestone: Type system for parameter value constraints with soundness proofs.**
Prerequisites: 5b

d. Develop **resource type systems for bytecode**. *(6 months)*
Deliverables: conference/journal paper *Prerequisites: 1a,2b,4a,4b,5a*
Here we want to invent low-level resource type systems for the bytecode itself. There

are several reasons to want to do this. First, because the type system will have a closer connection with the machine than the high-level type systems, we can have greater confidence that it truly reflects the resource usage of the machine (especially if later work in 6 and 8 draw on this). Moreover, it establishes a *common resource typing* level, that we might utilise in generalizing the high-level type systems to different languages and language constructs (for example in 7c).

There are also specific formal properties that a bytecode type system enables. A low-level type system makes it possible to state and prove a *type preservation property* for a compiler: that a well-typed high-level language compiles to well-typed bytecode. And we can formalize the *type preservation of bytecode optimisations* using a bytecode type system. In our setting, well-typed means that relevant resource bounds are met, and type preservation will mean that the *same* resource bounds are met.

Work on this subtask will include design of the type systems for the bytecode, perhaps adopting the ideas from 4b and 5a. It could also include formal proofs of soundness for the the bytecode logic, analogous to 4e and 4f.

This subtask is related to existing work on applying high-level programming language type systems to low-level assembly code [39, 17] and work on type systems for Java bytecode [51]. A more direct connection is with recent work on applying space-bound type systems to low-level assembly code [5].

e. **Implementation of the bytecode type systems** *(5 months)*
Deliverables: experimental implementation *Prerequisites: 2b,5d*
The task here is to implement the type system from above. The implementation will consist of a concrete way to represent bytecode typing annotations, along with a type-checker which verifies whether such an annotation is valid. This implementation could then form part of the integration with our chosen virtual machine, which will be undertaken in 8.

f. **Milestone: Implemented resource type system for bytecode.** *Prerequisites: 5e*

Deliverables
D5a, D5b, D5d, D5e

Milestones and expected result
Type system for parameter value constraints with soundness proofs
Implemented resource type system for bytecode

Workpackage description

Workpackage number: WP6 — Generation of certificates
Start date: month 14
Person-months for each participant: 11 pm for participant 1, 6 pm for participant 2

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

Description of work

This package is concerned with generating mobile guarantees of resource boundedness. The guarantee, or *certificate*, is what will be shipped together with the code, as irrefutable evidence for the consumer that the code obeys the desired resource constraints. Here we consider the format of the certificates, and their generation.

a. Implement a **certificate generator**. *(6 months)*
Deliverables: prototype implementation *Prerequisites: 4c*
Given a program which is typed in one of the high-level type systems developed in 4 and 5, we want to automatically generate a certificate which provides manifest evidence of this fact. The certificate contains a proof in our program logic. Here we must design a format for certificates (perhaps based on XML), and implement a software component which generates these from type-checked programs.

b. **Milestone: Certificate generator.** *Prerequisites: 6a*

c. Smaller certificates with **formalized soundness proof**. *(5 months)*
Deliverables: report on experimental implementations *Prerequisites: 6a,4f*
We expect that certificates which encode full proofs in our program logic may be too large to ship routinely with programs. Therefore we will look at ways of reducing the size of shipped certificates. One idea is to *formalise* the soundness proof in 4f, and ship this once only (or build it into the VM's security manager, see 8). Then instead of putting a full proof in a certificate, we just enclose a typing derivation in the type system: the full proof can be recreated in the client by plugging the typing derivation into the generic soundness proof and normalising, treating the soundness proof as a lemma. In this task, we want to experiment with this approach to test its feasibility.

d. Smaller certificates with **prooflets**. *(5 months)*
Deliverables: report on experimental implementations *Prerequisites: 6a,4f*
Another idea for reducing certificate size is to ship a "prooflet" — a small piece of code which can re-create the proof in the client, when (or if) it is called to do so. This is similar to the idea of a *tactic* in automated theorem proving. This novel suggestion requires some careful design, because of the resource implications for executing the prooflet itself. One solution would be to sandbox the prooflet, to ensure that it could not violate resource bounds. (Notice that the code consumer may well require a different set of resource bounds to be satisfied for the proof-checking phase).

e. **Milestone: Experiments with reducing certificate size.** *Prerequisites: 6c,6d*

f. Optional: implement a **component to emit bytecode typing**. *(1 month)*
Deliverables: prototype implementation *Prerequisites: 5d,4c*
Since we plan to have resource type systems also at the level of VM bytecodes according to 5d, then it will be a realistic alternative to enclose bytecode typing derivations in the resource certificates. Again, we can call on a formalized soundness proof for the bytecode typing to recreate complete proofs in the program logic. This is liable to be easier than the approach in 6c, since the bytecode is closer to the program logic. It also has the advantage that it is not tied to a particular high-level language and compiler. However, typing derivations in the bytecode will be much larger than those in the high-level language, and may be comparable to the size of the plain certificates in 6a. This could be carried out as a final year student project, and the manpower estimate is for supervision time.

The work in this package is mainly applied research, involving the design and construction of software components. Working software from this package will be an essential part of the final overall system.

Package 9 is dedicated to advanced research topics with the motivation of reducing certificate size. Parts 6c and 6d in this workpackage address this issue. We expect these approaches to be fruitful but not the final word; the results will be useful input for 9.

Deliverables
D6a, D6c, D6d, D6f

Milestones and expected result
Certificate generator
Experiments with reducing certificate size

Workpackage description

Workpackage number: WP7 — Advances in high-level type systems Start date: month 10 Person-months for each participant: 6 pm for participant 1, 15 pm for participant 2

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

Description of work

The goal of this workpackage is to improve expressiveness, user-friendliness, accuracy of the type systems developed under 4 and 5.

a. Improving accuracy of type system by **allowing for user interaction**. *(3 months)*
Deliverables: technical report *Prerequisites: 4a,4b,5a*
The aim here is to augment the basic type system from 4 with user annotations in the form of supplied proofs based, for example, on the derived rules from 4a. Also more abstract annotations such as loop invariants could be considered here.

If very restrictive resource bounds are imposed (e.g. hard limits on stack size) we might have to give the programmer the possibility to implement certain critical methods directly in bytecode with corresponding certificates obtained by hand, supported by a theorem prover. The task here will be to enable smooth integration of such user-supplied and -certified routines with high-level code.

b. **Adaptation** of bound generation/certification **to optimisations** *(5 months)*
Deliverables: research paper; prototype implementation *Prerequisites: 3e*
Usually, applicability of compiler optimisations (such as those mentioned in 3e) is decided on an ad-hoc basis with correctness of the optimisation being the only criterion. Whether or not an optimisation has taken place is not made visible to the user, it is only through improved overall runtime and space behaviour that the user becomes aware of them.

The purpose of this task is to identify static approximations as to the applicability of certain optimisations and to take their effect into account when calculating resource bounds and certificates. When a tail recursive definition is transformed into an iteration, no memory space for maintaining a stack should be counted when computing resource estimates. Similarly, we must report and appropriately account for the possibility for reusing a temporary file as opposed to creating a new one.

In order to be able to compute resource bounds statically and to retain transparency for the user, it becomes necessary to delineate the applicability of an optimisation by well-defined static criteria. For instance, it will not be sufficient to implement optimisation of tail recursion by discarding the stack frame of the caller at runtime in case the called function is in tail position, because the effect of this on resource usage depends on dynamic aspects known only at runtime. We rather have to syntactically characterise tail recursion and transform the code before actually running it.

Similar but more challenging is the situation with garbage collection. In order to be able to take its effect into account we will have to consider static approximations such as the type system in [31].

This substantially extends the approach to compiler optimisation in [41, 49] where it is shown how correctness proofs for an unoptimised program can be transformed into proofs for the optimised program by composing with a general proof that the optimisation is semantics preserving.

In order to demonstrate feasibility it will be sufficient to restrict attention to two representative optimisations, for example tail recursion as iteration and static memory reuse in the style of [31].

c. **Extend** basic type system **to object-oriented language** *(6 months)*
Deliverables: research paper; extension of implementation *Prerequisites: 3d,4b,5a*
The basic type system developed in 4 will be likely to encompass only the first-order side effect-free fragment of our high-level language. The aim here is to generalise to account for object-orientation. While we may be able to draw inspiration from encodings of object-oriented languages in functional ones [13], genuine innovation is required for two reasons. Firstly, these encodings invariable rely on advanced features such as higher-order functions and (mixed-variance) recursive datatypes; secondly the compilation of object-oriented features will make use of the object-oriented structure of the VM rather than following an encoding. Our strategy will be to extend the previous work to encompass higher-order functions (perhaps restricted to linear [54, 29]) and then tackle object-orientation proper.

Another issue that might arise would be an extension of the previous work to more object-specific resources such as "number of class and interface loads required".

d. **Type inference** *(6 months)*
Deliverables: research paper; implementation (optional) *Prerequisites: 4b,5a,7a,7b*
The basic type system developed under 4 and the extensions developed under 7a and 7b may rely on any number of user-supplied type annotations, for instance, recursive functions might be annotated by suggested bounds on their resource usage which are merely certified, cf. [18].

The aim here is to develop ways to infer such annotations automatically to a certain degree based on decision procedures for arithmetic inequalities [32], automata-theoretic methods [47], unification [37], program analyses, fixpoint methods [46], etc.

e. **Extend** basic type system **to mutable state and concurrency** *(1 month)*
Deliverables: technical report *Prerequisites: 3f*
We will merely assess and elaborate the requirements for this extension possibly leading to future work.

This workpackage forms part of the scientific core of the proposal. Successful completion will demonstrate that our proposal extends beyond the basic feasibility validated in 1, 2, 4. The goals set out here are ambitious but realistic. In case of difficulty or progress slower than expected it is possible to scale down by e.g. dropping the parts of 7b related to memory management.

Deliverables
D7a, D7b, D7c, D7d, D7e

Milestones and expected result

Workpackage description

Workpackage number: WP8 — Integration with existing security model
Start date: month 27
Person-months for each participant: 6 pm for participant 1, 2 pm for participant 2

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

Description of work

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

a. **Relationship with existing security management** *(4 months)*
Deliverables: technical report *Prerequisites: 5e,6a*

One direct advantage of a security infrastructure based on certificates and proof would be the ability to safely disengage the existing security manager for any downloaded code which can be shown not to offer any potential threat to the host cf. [20]. Thus a *dynamic* (run-time) security management overhead would be replaced by a *static* (load-time) security assessment cost for those mobile code routines whose certificate guarantees that it satisfies the resource requirements of the host. Where the attached certificate *cannot* provide this guarantee (or cannot *be shown to* provide this guarantee) then either all or some parts of the existing dynamic security management code will be needed to supplement the static security assessment.

b. **Experimental implementation** *(2 months)*
Deliverables: experimental prototype *Prerequisites: 2c,8a,5e,6a*

A prototype implementation of a certificate-led resource manager as outlined in Section 5.1 will be produced. This will provide a platform for further speculative research on developments in static security assessment. Recent work on Java-based agent models which work with the Java 2 security model [22] would provide the basis for further development here.

c. **Milestone: Implemented resource manager.** *Prerequisites: 8b*

d. **Resource typing of native methods** *(2 months)*
Deliverables: operational techniques *Prerequisites: 1b,5e*

It will be necessary to have a least a conservative estimate of the likely cost of invoking the native methods of the virtual machine (that is, those methods which have no bytecode representation). We will develop estimation techniques for this purpose.

Deliverables
D8a, D8b, D8d

Milestones and expected result
Implemented resouce manager

Workpackage description

Workpackage number: WP9 — Reducing size of certificates; negotiation vs. proof
Start date: month 25
Person-months for each participant: 9 pm for participant 1, 9 pm for participant 2

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

Description of work

This package is concerned with exploring possible alternatives in situations where the generation and transmission of 100%-guaranteed certificates is unfeasible 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.

This workpackage is more tentative and visionary than the other ones. Progress will be strongly dependent on the results obtained in the other packages and the particular qualifications of the research assistants.

A minimum deliverable will consist of visionary articles fleshing out the ideas thus enabling future interaction and perhaps collaboration with others working on these issues.

a. Study the size-reducing effect of **proof-theoretic methods** *(4 months)*
Deliverables: research paper *Prerequisites: 2c,6c,6d,6f*

As previously mentioned we will enable the use of lemmas and tactics, low-level type systems for bytecode, and programmable tactics in order to reduce the size of certificates. This task is concerned with analysing these methods in view of their effect on size reduction and possible improve them. We always assume that certificates are being compressed so that certain size reductions performed on raw proofs might not actually show up as such. For instance, contracting non-parametrised definitions might not lead to any reduction after Ziv-Lempel encoding.

b. Interaction-based **probabilistic certification** *(7 months)*
Deliverables: research paper *Prerequisites: 6a*
Here we plan to depart from the requirement of endowing mobile code with mathematical proofs of resource bounds.

The general scenario will be as follows: A sends B a piece of code to be executed remotely. On the basis of the code and possibly random data B computes a challenge (in the style of "please give letters 3 and 7 of your online password") to which A must respond. B may then accept the code, reject it, or set another challenge.

The theory of *probabilistically checkable proofs* [3, 4] shows how based on a concept of polynomially-sized and polynomial time verifiable certificate (which we can assume here) such a protocol can be devised so that challenges have logarithmic size, responses have constant size, and the probability that A can give a correct response to a challenge although no certificate exists is $<50\%$ so that k independent rounds lead to an error probability of $\leq 2^{-k}$.

In spite of some encouraging recent work [27] the overhead on the side of A remains considerable. We plan to investigate feasibility of this approach in our context and study possible relaxations, for instance allow for larger size responses. We emphasize that our concern is not to advance the theory of probabilistically checkable proofs, but exclusively to harness it for the purpose of certification of mobile code.

c. **Negotiation** in the absence of rigorous proofs *(7 months)*
Deliverables: visionary paper; prototype implementation (optional) *Prerequisites: 2a,2b*
The above-described protocol still requires that the sender A actually possesses a proof that his program meets the required bounds. It only reduces the amount of information that is actually interchanged. For the case where proofs are too difficult or impossible to obtain, we propose to investigate more liberal negotiation processes like the following:

- B provides a range of input parameters, A sends certificate which works only for this range.
- B challenges specific parts of the program. E.g. "you've got a write command in line XXX. Please convince me that this is fine." A then responds with a proof with assumptions that B may challenge again or accept.
- To cope with resource usage depending on unpredictable extraneous factors such as interaction patterns in concurrent systems or number of cache misses, we propose to investigate the use of probabilistic models such as PEPA [21, 28]. A and B would agree upon a probabilistic model to be used; A would then carry out the modelling and would provide B with verifiable results obtained in this way, for instance in the form of probability matrices or selected rows/columns thereof.

This package is appreciably more risky than the previous ones as we move further away from well-understood terrain in programming language theory. However, the possible reward will be high, as size of certificates and overhead in their production forms the biggest foreseeable obstacle against wide practical use of resource certification. If it can be successfully overcome or at least the necessary foundations laid, we will have paved the way for practical resource certification in the context of global computing.

Deliverables
D9a, D9b, D9c

Milestones and expected result

Workpackage description

Workpackage number: WP10 — Mobile virtual machines Start date: month 4 Person-months for each participant: 14 pm for participant 1, 2 pm for participant 2

Objectives Investigate extension to support downloadable virtual machines

Description of work

This workpackage develops a thread of investigation into a mechanism to support mobility between computational environments. As with 9 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. Mobile virtual machines can be realised as *circlets* [12]. 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.

a. Understanding the **practical effectiveness** of the technology *(4 months)*
Deliverables: internal technical report *Prerequisites: 1b*

To add another layer of software interpretation to the static virtual machine model calls into question the practical usefulness of this technology in terms of system performance. To consider that the target platform of the mobile virtual machine might be a handheld device further heightens this concern. This task will investigate the use of re-configurable hardware as an implementation technology for this concept. Reconfigurable hardware offers the promise of performance close to that of circuitry but without the same specificity.

b. A **metalanguage for virtual machines** *(6 months)*
Deliverables: research paper *Prerequisites: 10a,4b,5d*

Another tool to provide partial support for this technology would be a configuration language (or metalanguage) for describing the configuration of next-generation configurable virtual machines. These VMs could then be subject to *just-in-time performance tuning* just before bytecode interpretation by setting or disabling certain optimisation methods. Recent developments in Java technology such as the Java 2 Micro Edition (J2ME) release already define the notion of a *configuration* as a virtual machine and a minimal set of core class libraries and APIs. A J2ME configuration specifies a generalized runtime environment for consumer electronic and embedded devices. Our configuration language would extend this through reference to our virtual machine cost model.

c. **Relationship to resource boundedness** *(6 months)*
Deliverables: research paper *Prerequisites: 10b*

Certain optimisations which are enabled by some virtual machines (and not by others) will have an effect which is still significant at the level of our abstract cost model of the virtual machine. In this study we will develop a cost model for mobile virtual machines. This could be developed from a series of simpler models which are then revised and extended. Particular examples of simple virtual machines which could be used in this study include a classical JVM and a CVM. The CVM is a full-featured, Java 2 virtual machine targetted for the next generation of consumer electronic and embedded devices. Typically, these devices run a 32-bit microprocessor/controller and have more than 2.0Mb of total memory for the storage of the C virtual machine and libraries.

Deliverables
D10a, D10b, D10c

Milestones and expected result

Workpackage description

Workpackage number: WP11 — Project management, dissemination and evaluation Start date: month 0 Person-months for each participant: 1 pm for participant 1, 0 pm for participant 2

Objectives Project management, dissemination and evaluation

Description of work

The project requires close collaboration between the two sites and provides many opportunities for dissemination. The purpose of this workpackage is to ensure that collaboration proceeds effectively and with attention to internal and external evaluation, while being able to take advantage of a wide variety of forms of dissemination for the results.

Project management plans are described in Section 9.7. The small size of the project enables decisions about the overall technical direction of the project to be taken in close consultation with all of the people involved. Milestones and periodic meetings provide checkpoints where progress can be reviewed and plans adjusted if necessary. Meetings for technical coordination will be as follows:

- A kickoff workshop in month 2 plus workshops in months 11, 23 and 35. These will be attended by all project personnel, insofar as possible. One prominent non-EU expert will be invited to speak at each of the first three workshops at the project's expense; this will give a useful source of comment and advice without the need for project staff to visit these people individually. All of these workshops will be open to people from outside the project, with the final workshop being publicized more widely.

- Internal project meetings in months 7, 17 and 29. Each of these will include technical meetings on all active workpackages, and will be attended by all project personnel involved with those workpackages.

Individual visits are also planned for collaborative technical work.

The results of the project will be made available to all through a website set up at the beginning of the project and maintained under the direction of the Project Coordinator for the duration of the work. This is intended to give interested parties a view of the results as they accumulate. It will include at least the following:

- An introduction to the project including title, partners, and summary, with links to appropriate European Commission websites (GC, FET, IST and/or FP5).

- All of the deliverables and other publications produced by the project, as they are produced. Those that are most appropriate for external consumption will be given special prominence.

- A section (protected from access by non-project personnel) for working drafts, internal project documents, etc.

The results of the project will also be presented at appropriate conferences and published in academic journals. Many of these conferences take place outside the EU (see Section 11) and this is taken into account in the travel budget.

The project workshops provide an opportunity for external participants to learn about the project's progress and to contribute their views. The final workshop will be associated with an established international conference for increased visibility; this event is intended more for disseminating the results of the project than for technical coordination and a proceedings is planned. Linking workshops with the annual project evaluation meetings will allow more efficient use of the travel budget.

For self-assessment, each Workpackage Coordinator (see the list in Section 9.7.1) will supply in advance measurable criteria of progress/success for the different stages of the workpackage which will later be used to assess progress. This assessment of progress will take place in connection with each of the end-of-year project workshops.

a. **Project website** *(1 month)*
Deliverables: website

b. **Kickoff workshop** *(0 months)*
Deliverables: workshop

c. **Workshop at end of year 1** *(0 months)*
Deliverables: workshop

d. **Workshop at end of year 2** *(0 months)*
Deliverables: workshop

e. **Workshop at end of year 3** *(0 months)*
Deliverables: workshop, proceedings

f. **Measurable criteria of progress/success** *(0 months)*
Deliverables: report

g. **Assessment of progress in year 1** *(0 months)*
Deliverables: report

h. **Assessment of progress in year 2** *(0 months)*
Deliverables: report

i. **Final assessment of progress** *(0 months)*
Deliverables: report

j. **Dissemination and use plan** *(0 months)*
Deliverables: report

k. **Technological implementation plan** *(0 months)*
Deliverables: report

Most of the deliverables are allocated 0 person-months because this work will be done by the main investigators rather than the researchers who are employed by the project.

Deliverables
D11a, D11b, D11c, D11d, D11e, D11f, D11g, D11h, D11i, D11j, D11k

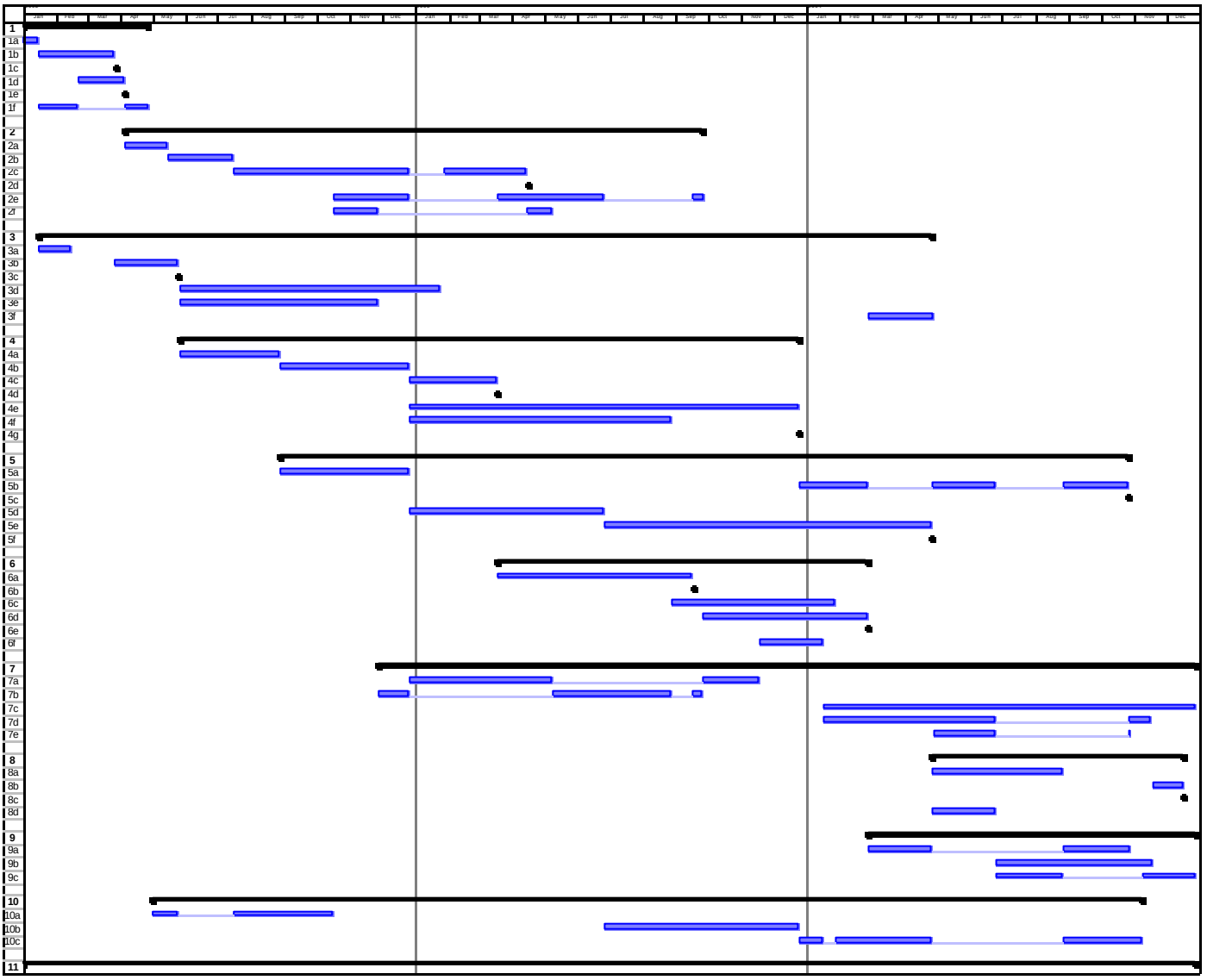
Milestones and expected result

9.4 Deliverables list

Del. no.	Deliverable name	WP no.	Lead participant	Est. person-months	Del. type	Security	Delivery (proj. month)
D1a	Definition of virtual machine platform	1	1	2	report	Pub	0
D1b	Cost model	1	1	6	report	Pub	2
D1d	Comparison of JVML with .NET for project use	1	1	3	report	Pub	3
D1f	Representative examples for project use	1	1	2	report	Pub	3
D2a	Language of assertions for bytecode logic	2	1	2	report	Pub	4
D2b	Proof rules for bytecode logic	2	1	3	report	Pub	6
D2c	Proof checker for bytecode logic	2	1	4	prototype	Pub	15
D2e	Theorem prover for bytecode logic	2	1	3	prototype	Pub	20
D2f	Encoding of VM semantics in theorem prover [optional task]	2	1	1	prototype	Pub	16
D3a	Definition of experimental high-level language	3	1	1	report	Pub	1
D3b	Compiler for high-level language	3	1	4	prototype	Pub	4
D3d	Extend compiler with immutable objects and higher-order functions	3	1	4	prototype	Pub	12
D3e	Extend compiler with optimisations	3	1	3	prototype	Pub	10
D3f	Extend with mutable state and concurrency [optional task]	3	1	1	prototype	Pub	27
D4a	Reasoning principles for resource usage	4	1	6	report	Pub	7
D4b	Type system for space-like resources	4	1	4	report	Pub	11
D4c	Typechecker for compiler	4	1	4	prototype	Pub	14
D4e	Proof of soundness over cost model	4	1	6	report	Pub	23
D4f	Proof of soundness over bytecode logic	4	1	4	report	Pub	19
D5a	Type system for expressing limits on parameter values	5	1	4	report	Pub	11
D5b	Proof of soundness for parameter value constraints	5	1	3	report	Pub	33
D5d	Resource type system for bytecode	5	1	6	report	Pub	17

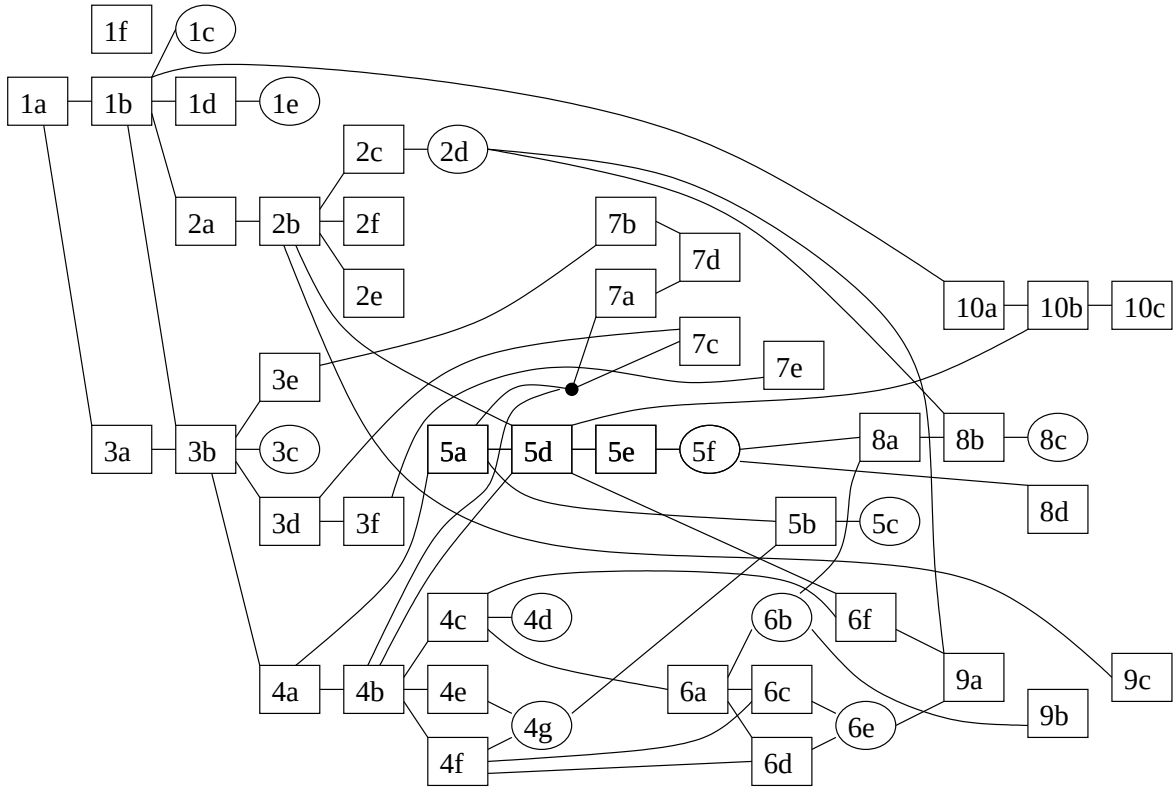
Del. no.	Deliverable name	WP no.	Lead participant	Est. person-months	Del. type	Security	Delivery (proj. month)
D5e	Implementation of bytecode type system	5	1	5	prototype	Pub	27
D6a	Certificate generator	6	1	6	prototype	Pub	20
D6c	Experiment with smaller certificates via formalised soundness proof	6	1	5	report	Pub	24
D6d	Experiment with smaller certificates via prooflets	6	1	5	report	Pub	25
D6f	Bytecode typing derivation generator [optional task]	6	1	1	prototype	Pub	24
D7a	Extension of type system by allowing user annotations	7	2	3	report	Pub	22
D7b	Adaptation of bound generation/certification to optimisations	7	2	5	prototype, report	Pub	20
D7c	Extension of basic type system to object-oriented language	7	2	6	prototype, report	Pub	35
D7d	Methods to infer type annotations automatically	7	2	6	report, maybe prototype	Pub	34
D7e	Extension of basic type system to mutable state and concurrency	7	2	1	report	Pub	33
D8a	Relationship between proof-based certificates and present-day security management	8	1	4	report	Pub	31
D8b	Certificate-based resource manager	8	1	2	prototype	Pub	35
D8d	Methods for estimating cost of native methods	8	1	2	report	Pub	29
D9a	Size-reducing effect of proof-theoretic methods	9	2	4	report	Pub	33
D9b	Feasibility of probabilistic certification	9	2	7	report	Pub	34
D9c	Negotiation-based protocols for resource certification	9	2	7	report, maybe prototype	Pub	35
D10a	Practical effectiveness of mobile virtual machines	10	1	4	report	Pub	9
D10b	Metalanguage for describing virtual machine configuration	10	1	6	report	Pub	23
D10c	Cost model for mobile virtual machines	10	1	6	report	Pub	34

Del. no.	Deliverable name	WP no.	Lead participant	Est. person-months	Del. type	Security	Delivery (proj. month)
D11a	Project website	11	1	1	website	Pub	3
D11b	Kickoff workshop	11	1	0	workshop	Pub	2
D11c	Workshop at end of year 1	11	1	0	workshop	Pub	11
D11d	Workshop at end of year 2	11	1	0	workshop	Pub	23
D11e	Workshop at end of year 3	11	1	0	workshop, proceedings	Pub	35
D11f	Measurable criteria of progress/success	11	1	0	report	Pub	6
D11g	Assessment of progress in year 1	11	1	0	report	Pub	12
D11h	Assessment of progress in year 2	11	1	0	report	Pub	24
D11i	Final assessment of progress	11	1	0	report	Pub	36
D11j	Dissemination and use plan	11	1	0	report	Pub	6
D11k	Technological implementation plan	11	1	0	report	Pub	36



9.6 Graphical presentation of project components

The following diagram shows the dependencies between the technical tasks in the project. Workpackage 11 continues throughout the project with connections to all other tasks.



In many cases, dependencies are not absolute — work on the successor task may commence before work on the predecessor task is completely finished.

Task 1f feeds into many subsequent tasks; to avoid clutter these dependencies are not shown in the diagram.

Milestones (drawn as circles) represent important decision points and completion of major components of the project. If any milestone is not achieved on time, additional effort will be diverted to it at the possible expense of later tasks. We hope to anticipate such problems (as part of Workpackage 11) so as to avoid slippage in the schedule.

The project plan is based on the use of JVML (task 1a) but we will investigate an alternative, .NET (task 1d). If we switch to this (milestone 1e) then a month or two may be lost but no more because the two platforms have many similarities.

9.7 Project management

For a project of this size, a light-weight management structure suffices. We separate the description of technical coordination from administrative/financial management although the structure is essentially the same for both.

9.7.1 Technical coordination

The Project Coordinator, Don Sannella (Edinburgh), is in overall charge of the technical work in the project. He is responsible for overseeing progress on the various workpackages and for initiating remedial action in case technical difficulties in some area necessitates a rethink of the overall project structure. He will convene, and be advised by, a steering committee consisting of all the workpackage coordinators and supplemented by additional members as needs demand. This committee will convene quarterly, either in person or by video/teleconferencing. The main means of access to the deliverables of the project and to internal technical documents will be via WWW, and the Project Coordinator is responsible for setting up and maintaining the project's Web site. The Project Coordinator is also responsible for organising the project's annual workshop.

Each workpackage is managed by a Workpackage Coordinator. These are as follows:

Workpackage 1 : Don Sannella (Edinburgh)

Workpackage 2 : Don Sannella (Edinburgh)

Workpackage 3 : Ian Stark (Edinburgh)

Workpackage 4 : David Aspinall (Edinburgh)

Workpackage 5 : Ian Stark (Edinburgh)

Workpackage 6 : David Aspinall (Edinburgh)

Workpackage 7 : Martin Hofmann (Munich)

Workpackage 8 : Stephen Gilmore (Edinburgh)

Workpackage 9 : Martin Hofmann (Munich)

Workpackage 10 : Stephen Gilmore (Edinburgh)

Workpackage 11 : Don Sannella (Edinburgh)

Each Workpackage Coordinator is responsible for monitoring the technical work within that workpackage and ensuring that it proceeds according to plan, convening meetings, maintaining an electronic mailing list for discussion on matters relevant to the technical work, reviewing internal technical documents and deliverables and maintaining a section of the project's website.

Decisions concerning the overall technical direction of the project will be made by the Project Coordinator in close consultation with the Workpackage Coordinators and other colleagues. A formal technical management meeting will be convened by the Project Coordinator at each of the project workshops and will be open to all project personnel. Project workshops will consist partly of presentations of technical developments and partly of a review of the progress of the project, and decisions concerning the future direction of the project will be informed by these reviews.

9.7.2 Administrative and financial management

The Project Coordinator represents the project externally and is responsible for liaising with the European Commission. He is responsible for allocating the project's resources to the various workpackages, for infrastructure at the various sites, and for the project's annual workshops. This allocation will be reviewed periodically and when technical plans change, and adjustments to the allocations may be made when this is deemed beneficial to the overall activity. Such decisions are made in close consultation with the Workpackage Coordinators.

Each Workpackage Coordinator manages the expenditure of resources allocated to his workpackage to employ researchers and to fund working meetings and other travel. Decisions here are taken in close consultation with the other principal researchers involved with that workpackage.

A 50% position for a senior researcher has been included in the budget. It is planned that this will be filled by the Project Coordinator who will be released from 50% of his normal duties to devote this time to technical work on the project, provided this is acceptable under Commission rules, while managing the research as part of his normal duties. Alternatively, this funding will be used to bring a senior researcher to the Edinburgh site for 18 months. In this case, the Project Coordinator will manage the project as described above and will assist with the research to the extent that his normal duties permit.

10 Clustering

There seems to be potential for exchange at some level between MRG and most GC-funded projects. However, we believe that the best opportunity for fruitful scientific interchange is clearly with projects where type systems play a central role, as in MRG (see workpackages 4, 5 and 7). These projects are: DART, MIKADO, MYTHS and PROFUNDIS. Possibilities for cross-fertilization include common workshops and invitation of observers to project workshops.

We would also be interested to have some MRG workshops and project meetings in common with other GC-funded projects to reduce the overhead of organizing meetings. This makes most sense for projects having partners in Edinburgh (DART, DEGAS) or Munich (AGILE) since these are the planned locations of MRG workshops and meetings.

For increased visibility, we would like to join the MRG final workshop with other GC-funded project workshops. A single event for all these projects would probably be too large to be practical; one possibility is to organize several workshops, with one involving the projects in the "types" cluster suggested above.

11 Other contractual conditions

Although the majority of work within the project can be done within the EU, there will be occasions when group members must travel outside the EU for study visits or for dissemination of results. We intend to present our work at appropriate leading international conferences, many of which often take place outside the EU. These include the IEEE Symposium on Logic in Computer Science (LICS), the ACM Symposium on Theory of Computing (STOC), the IEEE Symposium on Foundations of Computer Science (FOCS), the ACM Conference on Programming Language Design and Implementation (PLDI), the ACM Symposium on Principles of Programming Languages (POPL) the International Conference on Functional Programming (ICFP), the IFIP International Conference on Formal Methods for Open Object-Based Distributed Systems (FMOODS), and the ACM Conference on Measurement and Modeling of Computer Systems (SIGMETRICS), all of which are usually held in the US. Other relevant leading international conferences which take place outside the EU are the International Symposium on Theoretical Aspects of Computer Software (TACS) which is held in Japan, and Foundations of Software Technology and Theoretical Computer Science (FSTTCS) which is held in India.

Most of the internationally leading relevant work in this area is being done in the US. Several trips are planned in combination with travel to conferences, with visits to as many of the following people as practical: Andrew Appel at Princeton, Peter Lee at Carnegie Mellon, George Necula at UC Berkeley (proof-carrying code); Karl Crary at Carnegie Mellon, Daniel Leivant at the University of Indiana, John Mitchell at Stanford, Andre Scedrov at the University of Pennsylvania, Stephanie Weirich at Cornell (resource bounds); John Mitchell at Stanford, Alessandro Coglio, Allen Goldberg and Zhenyu Qian at Kestrel Institute (JVM formalisation); Benjamin Pierce at the University of Pennsylvania (collaborator with Hofmann on types in object-oriented languages); Dilsun Kırlı at MIT (collaborator with Gilmore on types for mobile code and with Aspinall on types for resource bounds); Adriana Compagnoni at Stevens Institute (collaborator with Aspinall on types for resource bounds). A few of these people will be invited to participate in project workshops at the project's expense; this boosts the visibility of the workshops while saving project staff the effort and expense of visiting these key people individually.

Overall, travel plans are as follows.

- For project workshops: 12 trips of 4 days for Edinburgh personnel, for a total cost of 12000 euros; 9 trips of 4 days for Munich personnel, for a total cost of 9000 euros; and 3 trips of 7 days for non-EU experts, for a total cost of 4080 euros.
- For internal project meetings: 4 trips of 4 days for Edinburgh personnel, for a total cost of 4000 euros; and 6 trips of 4 days for Munich personnel, for a total cost of 6000 euros.
- For individual visits to the other partner for collaborative work: 9 trips of 7 days for Edinburgh personnel, for a total cost of 12240 euros; and 6 trips of 7 days for Munich personnel, for a total cost of 8160 euros.
- For conference attendance: 18 trips of 4 days for Edinburgh personnel, for a total cost of 28200 euros; and 9 trips of 4 days for Munich personnel, for a total cost of 15600 euros. It is anticipated that two of these per year (one per site) will be to non-EU conferences.

Prior approval will be sought from the Project Officer for all non-EU travel and invitees.

References

[1] Andrew W. Appel and Edward W. Felten. Proof-carrying authentication. In *Proceedings of the 6th ACM Conference on Computer and Communications Security*, November 1999.

[2] Andrew W. Appel and Amy P. Felty. A semantic model of types and machine instructions for proof-carrying code. In *Proceedings of the 27th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL '00)*, pages 243–253, January 2000.

[3] S. Arora, R. Motwani, M. Safra, M. Sudan, and M. Szegedy. Proof verification and intractabilitity of approximation problems. In *Proc. 33rd IEEE Symp. on Foundations of Computer Science*, pages 13–22, 1992.

[4] Sanjeev Arora. *Probabilistic Checking of Proofs and Hardness of Approximation Problems*. PhD thesis, UC Berkeley, 1994. UCB Technical Report: CS-TR-476-94.

[5] David Aspinall and Adriana Compagnoni. Heap bounded assembly language. Submitted, 2001.

[6] Lennart Augustsson. Cayenne — a language with dependent types. In *ICFP '98: Proceedings of the Third ACM SIGPLAN International Conference on Functional Programming*, pages 239–250. ACM Press, 1998.

[7] S. Bellantoni, K.-H. Niggl, and H. Schwichtenberg. Ramification, Modality, and Linearity in Higher Type Recursion. *Annals of Pure and Applied Logic*, 2000. to appear.

[8] Stephen Bellantoni and Stephen Cook. New recursion-theoretic characterization of the polytime functions. *Computational Complexity*, 2:97–110, 1992.

[9] G. Bellè and E. Moggi. Type intermediate languages for shape-analysis. In *Typed Lambda Calculi and Applications: Proceedings of the Third International Conference TLCA '97*, number 1210 in Lecture Notes in Computer Science, pages 11–29. Springer-Verlag, April 1997.

[10] P. Nick Benton, Andrew Kennedy, and George Russell. Compiling Standard ML to Java bytecodes. In *ICFP '98: Proceedings of the Third ACM SIGPLAN International Conference on Functional Programming*. ACM Press, 1998.

[11] Ralph Benzinger. Automatic complexity analysis revisited. Slides for a talk at Cornell University. See `http://www.cs.cornell.edu/Nuprl/PRLSeminar/PRLSeminar99_00/benzinger/ja%n31.html`, January 2000.

[12] Gordon Brebner. Circlets: Circuits as applets. In *IEEE Symposium on FPGAs for Custom Computing Machines*, April 1998. Napa Valley, California.

[13] Kim B. Bruce, Luca Cardelli, and Benjamin C. Pierce. Comparing object encodings. In *Theoretical Aspects of Computer Software (TACS), Sendai, Japan*, September 1997. An earlier version was presented as an invited lecture at the Third International Workshop on Foundations of Object Oriented Languages (FOOL 3), July 1996.

[14] A. Coglio, A. Goldberg, and Z. Qian. Toward a provably-correct implementation of the JVM bytecode verifier, 1998.

[15] Christopher Colby, Peter Lee, and George C. Necula. A Proof-Carrying Code architecture for Java. In *Proceedings of the 12th International Conference on Computer Aided Verification (CAV00), Chicago*, 2000.

[16] Adriana Compagnoni. CAREER: A formally verified environment for the production of secure software. NSF project based at Stevens Institute of Technology, Hoboken, New Jersey. See `http://guinness.cs.stevens-tech.edu/~abc/` for more information., 2000.

[17] K. Crary, N. Glew, D. Grossman, R. Samuels, F. Smith, D. Walker, S. Weirich, and S. Zdancewic. TALx86: A realistic typed assembly language. In *1999 ACM SIGPLAN Workshop on Compiler Support for System Software Atlanta, GA, USA*, pages 25–35, May 1999.

[18] K. Crary and S. Weirich. Resource bound certification. In *Proc. 27th Symp. Principles of Prog. Lang. (POPL)*, pages 184–198. ACM, 2000.

[19] S. Freund and J. Mitchell. A formal framework for the Java bytecode language and verifier, 1999.

[20] Stephen Gilmore. Deep type inference for mobile functions. In P. Trinder G. Michaelson and H.-W. Loidl, editors, *Trends in Functional Programming (Volume 1)*, pages 40–48, 2000.

[21] Stephen Gilmore and Jane Hillston. The PEPA workbench: A tool to support a process algebra-based approach to performance modelling. In *Proceedings of the Seventh International Conference on Modelling Techniques and Tools for Computer Performance Evaluation, Springer LNCS vol. 794*, pages 353–368, 1994.

[22] Stephen Gilmore and Marco Palomino. BabylonLite: Improvements to a Java-based distributed object system. Submitted to *Concurrency and Computation: Practice and Experience*, March 2001.

[23] Algorithmic Solutions Software GmbH. LEDA — Library of Efficient Data types and Algorithms. A C++ class library originally developed at the Max-Planck-Institut für Informatik. See `http://www.algorithmic-solutions.com/as_html/products/products.html`.

[24] A. Gordon and D. Syme. Typing a multi-language intermediate code. Technical Report MSR TR 2000-106, Microsoft Research, 2000. A shorter version appears in Proceedings of Symposium on Principles of Programming Languages, (POPL2001), London, 2001.

[25] GSF. The diabcard project. `http://www-mi.gsf.de/diabcard/`, 2000.

[26] R. Harper, F. Honsell, and G. Plotkin. A framework for defining logics. *JACM*, 40(1):143–184, 1993.

[27] Prahladh Harsha and Madhu Sudan. Small pcps with low query complexity. *Electronic Colloquium on Computational Complexity*, 2000. Report No. 61.

[28] Jane Hillston. *A Compositional Approach to Performance Modelling*. Cambridge University Press, 1996.

[29] Martin Hofmann. Linear types and non size-increasing polynomial time computation. To appear in *Theoretical Computer Science*. See `www.dcs.ed.ac.uk/home/papers/icc.ps.gz` for a draft. An extended abstract has appeared under the same title in Proc. Symp. Logic in Comp. Sci. (LICS) 1999, Trento, 2000.

[30] Martin Hofmann. Safe recursion with higher types and BCK-algebra. *Annals of Pure and Applied Logic*, 2000. to appear.

[31] Martin Hofmann. A type system for bounded space and functional in-place update. In G. Smolka, editor, *Programming Languages and Systems*, pages 165–179. Springer LNCS, 2000.

[32] J. Hughes and L. Pareto. Recursion and dynamic data structures in bounded space: towards embedded ML programming. In *Proc. International Conference on Functional Programming (ACM). Paris, September '99.*, pages 70–81, 1999.

[33] Pankaj Kakkar, Michael Hicks, Jonathan T. Moore, and Carl A. Gunter. Specifying the PLAN networking programming language. In *Higher Order Operational Techniques in Semantics*, volume 26 of *Electronic Notes in Theoretical Computer Science*. Elsevier, September 1999.

[34] J.-Y. Marion and J.-Y. Moyen. Efficient first-order functional program interpreter with time bound certifications. In *LPAR 2000*, Springer LNAI, 2000.

[35] James McKinna. *Deliverables: A Categorical Approach to Program Development in Type Theory*. PhD thesis, University of Edinburgh, 1992. Report CST-96-92.

[36] Sun Microsystems. JavaCard software distribution and manuals. See `http://java.sun.com/products/javacard/`.

[37] Robin Milner. A theory of type polymorphism in programming. *Journal of Computer and System Sciences*, 17:348–375, August 1978.

[38] G. Morrisett, D. Tarditi, P. Cheng, C. Stone, R. Harper, and P. Lee. The TIL/ML compiler: Performance and safety through types. 1996 Workshop on Compiler Support for Systems Software, January 1996.

[39] G. Morrisett, D. Walker, K. Crary, and N. Glew. From System F to typed assembly language. *ACM Transactions on Programming Languages and Systems,*, 21(3):528–569, May 1999.

[40] Alan Mycroft and Richard Sharp. A statically allocated parallel functional language. In *Automata, Languages and Programming*, pages 37–48, 2000.

[41] George Necula. Proof-carrying code. In *Proceedings of the ACM Symposium on Principles of Programming Languages*, 1997.

[42] George Necula. *Compiling with Proofs*. PhD thesis, Carnegie Mellon University, September 1998.

[43] George C. Necula and Peter Lee. Safe, untrusted agents using Proof-Carrying Code. In *LNCS 1419: Special Issue on Mobile Agent Security*. Springer, 1998. Abstract.

[44] T. Nipkow, D. von Oheimb, and C. Pusch. μJava: Embedding a programming language in a theorem prover. In F.L. Bauer and R. Steinbrüggen, editors, *Foundations of Secure Computation. Proc. Int. Summer School Marktoberdorf 1999*, pages 117–144. IOS Press, 2000.

[45] S. Owre, J. Rushby, and N. Shankar. PVS: a prototype verification system. In *Proc. 11th Intl. Conf. on Automated Deduction, Springer LNCS vol. 607*, pages 748–752, 1992.

[46] Jens Palsberg and Michael Schwartzbach. Object-oriented type inference. In *Proc. ACM Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA)*, pages 246–161, 1991.

[47] Jens Palsberg, Mitchell Wand, and Patrick O'Keefe. Type inference with non-structural subtyping. *Formal Aspects of Computing*, 9:49–67, 1997.

[48] Lawrence C. Paulson. *Isabelle — A Generic Theorem Prover*. Lecture Notes in Computer Science 828. Springer-Verlag, 1994. For latest materials, see `http://www.cl.cam.ac.uk/Research/HVG/Isabelle/`. A new Isabelle book is in preparation.

[49] Amir Pnueli, Michael Siegel, and Eli Singerman. Translation validation. In *Proc. of the 4th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'98), LNCS 1384*, pages 151–166, 1998.

[50] J. C. Reynolds. Syntactic control of interference. In *Proc. Fifth ACM Symp. on Princ. of Prog. Lang. (POPL)*, 1978.

[51] Raymie Stata and Martin Abadi. A type system for Java bytecode subroutines. In *ACM Transactions on Programming Languages and Systems 21*, volume 21(1), January 1999.

[52] Jean-Pierre Talpin and Pierre Jouvelot. The type and effect discipline. In *Seventh Annual IEEE Symposium on Logic in Computer Science, Santa Cruz, California*, pages 162–173, Los Alamitos, California, 1992. IEEE Computer Society Press.

[53] D. Tarditi, G. Morrisett, P. Cheng, C. Stone, R. Harper, and P. Lee. TIL: A type-directed optimizing compiler for ML. In *Proceedings of the 1996 ACM SIGPLAN Conference on Programming Language Design and Implementation*, SIGPLAN Notices, 31(6). ACM Press, June 1996.

[54] Philip Wadler. Linear types can change the world. In *TC 2 Working Conference on Programming Concepts and Methods (Preprint)*, pages 546–566, 1990.

[55] Hongwei Xi and Robert Harper. A dependently typed assembly language. Technical Report OGI-CSE-99-008, Oregon Graduate Institute of Science and Technology, July 1999.

A Description of the consortium

This project is ambitious, demanding a blend of different skills and expertise and requiring tight coordination between the components of different workpackages. To facilitate collaboration we have decided to limit the consortium to just two sites, with sufficient resource per site to build a substantial team. Each site has a leading international reputation in a subset of the expertise required by the project. Edinburgh is internationally acclaimed for its work on semantics, type theory, programming language design and implementation, concurrency, and formal methods, with substantial recent work on aspects of global computation. Munich is internationally known for its work on formal methods, programming language theory and software engineering, with the recent addition of Hofmann contributing world-class expertise in resource-bounded computation and type theory. Each of the participants have international reputations in areas relevant to the project, as outlined below.

We do not anticipate difficulties in collaborating and integrating our expertise since several of the participants in the project have collaborated on research before and all participants know each other well, having worked together while members of the Laboratory for Foundations of Computer Science in Edinburgh. Aspinall has collaborated with Hofmann on type systems for resource bounds, Aspinall has collaborated with Sannella on type systems for specification, and Hofmann has collaborated with Sannella on semantics of specifications.

A.1 Division of Informatics, University of Edinburgh

The Division of Informatics (`http://www.informatics.ed.ac.uk/`) was formed from the existing Departments of Artificial Intelligence, Cognitive Science, Computer Science and associated research institutes in 1998, with the aim of promoting innovative research into the structure, behaviour, and interactions of natural and artificial computational systems. In the last UK research assessment exercise it had by far the largest concentration of top (5) rated researchers in this area anywhere in the UK. It also attained the highest rating in the UK teaching assessment exercise. Its research activities take place within a number of research institutes, including the Laboratory for Foundations of Computer Science (LFCS), the institute involved in this project.

LFCS (`http://www.lfcs.informatics.ed.ac.uk/`) is an internationally renowned centre for research in theoretical computer science. Current LFCS research projects, which are carried out with the support of national and European funding, are on various topics within the following general areas: semantics and type theory; programming languages, specifications, and design; concurrency and distributed systems; complexity and analysis of algorithms. This work covers a wide spectrum, from basic research on foundations to collaborative projects with industry on applications. All of these projects benefit greatly from daily interaction with each other and with the excellent PhD students and continuous stream of prominent visitors which LFCS attracts from all over the world. The proposed project fits well into LFCS's research programme. It takes advantage of existing expertise in programming language design, global computation, type theory, theorem proving and semantics to address the challenges of the Global Computing proactive initiative.

David Aspinall

David Aspinall (`http://www.dcs.ed.ac.uk/~da/`) is a Lecturer with the Laboratory for Foundations of Computer Science in the Division of Informatics at the University of Edinburgh. He received his PhD in 1997 from the University of Edinburgh. During and since his PhD, he has worked on several topics related to this proposal, including fundamentals of type systems and recent work on resource bounded type systems as part of a UK EPSRC-funded project *Type Systems for Resource-Bounded Programming and Compilation* begun in May 2000 with Hofmann. Aspinall also brings considerable experience in building high-quality research prototypes; for the last three years he has led work on the *Proof General* generic control system for interactive proof assistants (see `http://www.proofgeneral.org`).

Stephen Gilmore

Stephen Gilmore (`http://www.dcs.ed.ac.uk/~stg/`) is a Senior Lecturer in Computer Science in the Division of Informatics at the University of Edinburgh. He received his PhD from the Queen's University of Belfast in 1990. His research interests include type systems for higher-order programming languages and the application of process algebras to describe stochastic processes which are performance models of computer systems. He has worked on extending the classical type systems of functional programming languages to detect more programmer errors at compile time or at run-time. He has a particular interest in the use of functional languages as mobile-code languages. He has published more than 20 papers in journals and conferences and has edited the proceedings of the Third International Workshop on Process Algebra and Performance Modelling, the FIREworks workshop on Language Constructs for Defining Features, and Trends in Functional Programming Volume 2. He is the joint Programme Committee chair for the joint PAPM-PROMBIV workshop this year. He was the Edinburgh site leader for the ESPRIT Framework 4 initiative Feature Integration in Requirements Engineering.

Donald Sannella

Donald Sannella (`http://www.dcs.ed.ac.uk/~dts/`) received a PhD in Computer Science from the University of Edinburgh in 1982 where he has worked ever since, being appointed Professor in 1998. His research interests include the design of algebraic specification languages and functional languages, mechanised reasoning, foundations for algebraic specification and formal software development, and applying these foundations to the practical development of modular software systems from specifications. He has published more than 50 papers in journals and international conferences and has held a series of grants since 1985 for research projects in the area of verification and formal development of programs including an EPSRC Advanced Fellowship in 1992–1997 and an RSE/SOEID Fellowship during 1998. His current grants include the EC-funded *Common Framework Initiative Working Group* (CoFI WG, finishing in April 2001) of which he is overall coordinator. He is editor-in-chief of *Theoretical Computer Science* (responsible for part B: Logic, Semantics and Theory of Programming) and chairman of the steering committee of the ETAPS conference series, and is on the Council of the European Association for Theoretical Computer Science. He was closely involved in the definition of the Global Computing proactive initiative.

Ian Stark

Ian Stark (`http://www.dcs.ed.ac.uk/~stark/`) is a Lecturer in Computer Science in the Division of Informatics at the University of Edinburgh. He received his PhD from the University of Cambridge in 1995, and subsequently worked in Pisa, where he held a Marie Curie fellowship, and in the BRICS research institute at the University of Aarhus. His research interests include foundational models of programming languages and computation; in particular types, concurrency, and the functional and object-oriented paradigms. He has over 10 published papers, most recently on a calculus for global management of local resources. This year he is on the programme committee for the international conference TACS '01 in Sendai, Japan. He currently holds an EPSRC grant on "Reasoning about Names and Identity in Programming Languages", and is Edinburgh coordinator of the EC-funded APPSEM working group. Previously he worked on the EC-funded EuroFoCS and CLICS-II projects.

A.2 Institut für Informatik, Ludwig-Maximilians-Universität München

This department has a long-standing record in formal methods, programming language theory, and software engineering (Wirsing, Hennicker). There are excellent contacts with the Logic Group (Schwichtenberg) in the Mathematics Department and the theorem proving group at the Technical University (Nipkow).

The Theoretical Computer Science group that Hofmann will join in Autumn 2001 currently comprises researchers in complexity theory (Johannsen), type theory (Matthes) and theorem proving (Backofen, Kahle). This will fruitfully complement the research to be carried out in the project.

Researchers at the department have held numerous nationally- and EC-funded research grants in the past; there are also fruitful contacts to industry, e.g. Siemens and BMW, that will be used in order to get feedback on and input to the project.

Martin Hofmann

Martin Hofmann received a PhD in Computer Science from Edinburgh University in 1995. From 1995 to 1998 he worked as a research assistant for the Technical University of Darmstadt; from 1998 to April 2001 he held a Lectureship within the Division of Informatics of the University of Edinburgh being promoted to Reader in October 1999. He acquired a *Habilitation* in Mathematics from TU Darmstadt in 2000. He took up a position as Associate Professor (C3) with the Department of Mathematics of TU Darmstadt in April 2001 and is taking up a position as full Professor in Informatik at LMU München in Autumn 2001.

His research interests include theory and design of functional and object-oriented programming languages, mechanised reasoning about programs and their specification. His most recent works were concerned with the design of type systems capable of bounding computational complexity, in particular space consumption of programs. He has published more than 25 papers in journals, refereed collections, and conference proceedings and has held an EPSRC research grant since May 2000. In Darmstadt he was local coordinator of the EC-funded working group TYPES and has been member of the EC-funded LOGSEM and APPSEM projects. He has served on the programme committee of several international conferences and workshops.

B Contract preparation forms

[Insert here Contract Preparation forms.]