European Cooperation in the field of Scientific and Technical Research
- COST -

Secretariat

Brussels, 30 November 2007

COST 256/07

MEMORANDUM OF UNDERSTANDING

Subject :	Memorandum of Understanding (MoU) for the implementation of a European Concerted Research Action designated as COST Action IC0701: Formal Verification of Object-Oriented Software

Delegations will find attached the Memorandum of Understanding for COST Action IC0701 as approved by the COST Committee of Senior Officials (CSO) at its 169th meeting on 15 - 16 November 2007.

MEMORANDUM OF UNDERSTANDING
for the implementation of a European Concerted Research Action
designated as

COST Action IC0701

FORMAL VERIFICATION OF OBJECT-ORIENTED SOFTWARE

The Parties to this Memorandum of Understanding, declaring their common intention to participate in the concerted Action referred to above and described in the Technical Annex to the Memorandum, have reached the following understanding:

1. The Action will be carried out in accordance with the provisions of document COST 299/06 "Rules and Procedures for Implementing COST Actions" (or in any new document amending or replacing it), the contents of which the Parties are fully aware of.

2. The main objective of the Action is to develop verification technology with the reach and power to assure dependability of object-oriented programs on industrial scale.

3. The economic dimension of the activities carried out under the Action has been estimated, on the basis of information available during the planning of the Action, at 10 million EUR in 2007 prices.

4. The Memorandum of Understanding will take effect on being accepted by at least five Parties.

5. The Memorandum of Understanding will remain in force for a period of four years calculated from the date of the first meeting of the Management Committee, unless the duration of the Action is modified according to the provisions of Chapter V of the document referred to in Point 1 above.

A. ABSTRACT AND KEYWORDS

Software is vital for modern society. The efficient development of correct and reliable software is of ever growing importance.

This Action will concentrate on program verification: the construction of logical proofs that programs are correct. Logic-based technologies for the formal description, construction, analysis, and validation of software can be expected to complement and partly replace traditional software engineering methods in the future.

Already, program verification methods have outgrown the area of academic case studies, and industry is showing serious interest. The logical next goal is the verification of industrial software products. Most programming languages used in industrial practice (such as Java, C++, and C#) are object-oriented. The Action will therefore focus on the verification of programs written in object-oriented languages and the particular problems this entails.

The goal of this Action is to co-ordinate the development of verification technology, to achieve reach and power needed to assure reliability of object-oriented programs on industrial scale.

Keywords: program verification, formal methods in software engineering, program logics, formal specification, program correctness

B. BACKGROUND

B.1 General background

Software is vital for modern society. It manages finances, regulates power generation and communications, controls aeroplanes, and processes security-critical information. Hence, the efficient development of dependable software is of ever growing importance.

Current estimates of the cost of software error to the world economy are in the region of a hundred billion Euro per year, and increasing. The cost is shared among the producers of software and its users. Even more resources are spent on avoiding errors and finding them by testing software. Even if a small percentage of errors can be avoided and a small percentage of the testing can be replaced by formal methods, the economic benefit would be enormous.

Technologies for the formal analysis and validation of software - mostly based on logics and formal reasoning - have matured and can be expected to complement and partly replace traditional software engineering methods. This Action will concentrate on methods and tools for program verification. Program verification is the construction of logical proofs that programs are correct for all possible inputs, i.e., that they satisfy a given formal specification of what is considered correct behaviour - a degree of certainty that mere program testing cannot achieve.

Currently, research in this field is fragmented into groups focussing on different verification technologies (software model checking, deductive verification, type inference) and, thus, on different fragments of programming languages, kinds of programs, and program properties. These groups are parts of larger, mostly disjoint communities (software engineering, programming

language theory, compiler construction, logic), each with their own scientific publications, conferences, etc. Their communication and co-operation needs to be improved by providing a new networking platform.

Recently, Sir Tony Hoare, one of the pioneers of formal program verification, has proposed the problem of a "verifying compiler", i.e., a fully automated program verification system that can handle programs of arbitrary size and complexity, as a Grand Challenge in computer science. His proposal has met great interest and initiated several meetings by researchers working in the field (some of them are among the participants of the Action). Work on the challenge suffers, however, of a lack of organisational structure and co-operation. To a large extend, the challenge overlaps with the (very) long-term goals of the Action, which is further evidence that better co-operation and interaction in the field of program verification is, on the one hand, urgently needed and, on the other hand, could be very fruitful.

Co-operation in program verification has great potential, since different groups face the same challenges independently of the employed technologies. Most important are: modularisation of large programs and their correctness proofs; verification of software for concurrent, mobile, and distributed systems; verification of customisable and reusable programs that use language features such as aspects, generics, and parametrisation.

Most programming languages used in industrial practice (such as Java, C++, and C#) are object-oriented. The Action will therefore focus on the verification of programs written in object-oriented languages and the particular problems this entails. The goal is the development of verification technology with the reach and power to assure reliability of object-oriented programs on industrial scale.

A COST Action is the best option and an ideal tool to achieve this goal, as the research of the groups planning to participate in the Action is already funded by European research programs and national funding agencies. Needed is additional funding for cooperation and exchange to overcome the fragmentation of the field, co-ordinate the research, and foster collaborations. The instruments of COST Actions, in particular working group meetings, conferences, and training schools, are ideal means to achieve these goals.

B.2 Current state of knowledge

B.2.1 Methods in Formal Verification

The main methods in formal verification are deductive verification (B.2.1.a), software model checking (B.2.1.b), and static program analysis (B.2.1.c). They differ in the quality and strength of the properties that can be verified, and in the degree of automation that can be achieved in verification tools.

B.2.1.a Deductive verification

Deductive software verification is characterised by three ingredients: first, target programs as well as the properties to be verified are represented as logical formulae that must be proven to be valid; second, validity is proven by deduction in a logic calculus; third, computer assistance is used for proof search and bookkeeping.

In contrast to static analysis and model checking it is possible to model the semantics of the target programming language precisely, i.e., without abstracting from unbounded data structures (integers, lists, trees, etc.) or unbounded programming constructs (loops, recursion). It is possible to formalise and prove very far-reaching properties of target programs.

This precision has a price, of course: even more so than in model checking, it is difficult to predict how much computing resources might be needed to complete a verification task. In general, the theoretical limitations implied by computability theory exclude a verification system that can automatically prove program properties over infinite structures.

B.2.1.b Software model checking

Model checking is based on using an abstraction of dynamic systems to obtain finite models of reasonable size, which then can be checked and analysed efficiently.

Tools for software model checking extract from a given program and a property to be checked a finite model that can then be fed into a automated model checking tool. The main problem with the approach, namely state-space explosion, is mitigated by using abstract interpretations and techniques from static program analysis. Another problem is over-generalisation (loops and recursive calls are usually ignored), which, in the best case, leads to false positives and lets errors pass undetected in the worst. Still, software model checking has provided invaluable insights into the working of modern programs.

B.2.1.c Static program analysis

With static program analysis, different kinds of properties can be analysed (e.g., information flow, control flow, reachability, points-to information) and different analysis techniques are available (type-based analysis, constraint-based analysis, abstract interpretation). Originally the focus of static program analysis has been on applying results for optimisations in compilers. Recently, applications to verification tasks have been on the agenda.

Static program analysis is a push-button technology, i.e., checks can be done fully automatically. This, however, requires abstractions and approximations, which may lead to false warnings. When these occur in great numbers, they greatly reduce the value of the analysis.

B.2.2 Current State of the Research Community

In scientific research laboratories, guarantees based on complete verification have been given for small microprocessors, operating systems, programming language compilers, communication protocols, large mathematical proofs, and even the essential kernels of the proof tools themselves.

Now, program verification methods have outgrown the area of academic case studies, and industry is showing serious interest. The emergence of the Formal Techniques Industrial Association (www.fortia.org) and the German Verisoft project (www.verisoft.de), which involves many industrial partners, may serve as evidence.

The logical next goal is now the verification of industrial software products. Most programming languages used in industrial practice (such as Java, C++, and C#) are object-oriented. Research therefore focuses on the verification of programs written in object-oriented languages and the particular problems this entails.

There already are a number of (fairly isolated) research projects on formal verification of object-oriented software, in which verification tools have been developed (most of the European research groups that have worked on or are currently working on these projects and tools are participants of the Action):

- ESC/Java2 (secure.ucd.ie/products/opensource/ESCJava2)
- LOOP (www.sos.cs.ru.nl/research/loop/main.html)
- KIV (www.informatik.uni-augsburg.de/lehrstuehle/swt/se/kiv)
- KeY (www.key-project.org)
- Jive (www.sct.ethz.ch/research/jive)
- Krakatoa (krakatoa.lri.fr)
- SPEC# (research.microsoft.com/specsharp)
- JACK (www-sop.inria.fr/everest/soft/Jack/jack.html)
- Bogor (bogor.projects.cis.ksu.edu)
- Java PathFinder (javapathfinder.sourceforge.net/)
- SLAM (research.microsoft.com/slam)

Unfortunately, this variety of tools supports the observed fragmentation in the field of object-oriented program verification. Even those tools that target programs written in the same language use different specification languages and concepts. They cover different subsets of the target programming languages; and they cannot exchange their output. On the other hand, there have been redundant parallel developments as different groups face the same challenges independently of the employed technologies.

Tool and method integration is an important prerequisite for the advancement in the field. It is, therefore, the topic of one of the Working Groups of the Action. The other three Working Groups will target problems and challenges that are common to the different tools and methods (see the following section).

B.2.3 The State of Knowledge in Focus Topics of the Action

Co-operation in program verification has great potential, since different groups face the same challenges independently of the employed technologies. Most important are: modularisation of large programs and their correctness proofs; verification of software for concurrent, mobile, and distributed systems; verification of customisable and reusable programs that use language features such as aspects, generics, and parametrisation.

B.2.3.a Customisable and reusable programs

Current versions of modern programming languages have mechanisms for reusing programs and adapting them to particular applications. The industry uses these mechanisms more and more in order to increase customisation and reuse of code, decrease costs, and shorten product cycles. In this scenario, an adaptable program constitutes a blueprint, which can be used to produce a whole series of customised software products.

The most prominent customisation mechanisms are aspects (injecting code at specified places in a program at compile time), generics (programs with variable, "generic", parts that are instantiated by the compiler or at runtime), and reflection (modifying the program in the process of it being executed).

While some advances have been made on specifying customisable programs (in particular, in the area of aspect orientation), little research has been done on their verification.

B.2.3.b Concurrency

Since concurrent programs are notoriously hard to test, a variety of techniques have been developed for their verification. As in the sequential case these techniques trade-off usability for the strength of guarantees that can be made. For instance, static program analysis tools aim to ensure absence of certain classes of errors (e.g., data races) in a fully automated manner. The price for the automation is that the given guarantees are not very strong, and these methods can produce false positives. Software model checking has been very successful as a systematic testing tool for concurrent applications. It can give strong guarantees, but in general only treats systems, where the number of processes (and the data involved) is bounded. Deductive verification frameworks do not suffer from these limitations, but require non-negligible user interaction to achieve this result.

As matters stand, all the systems in the different classes have different coverage in terms of the supported features of the programming language. What all approaches have in common are very similar efforts to better understand the nature of concurrent programs and the techniques that programmers use to create working applications.

B.2.3.c Modularisation

Modularisation is the process of separating a program into independent parts with clearly specified interfaces between them. Modularisation is the key approach to managing complexity in software design, implementation, and also verification. There is a broad consensus that the modularisation problem has not been satisfactorily solved by the programming languages used in practice today, such as Java, C#, and C++. Thus, development of modularisation concepts has become a task for verification experts. Several partial solutions have been proposed for this problem, including different ownership control schemes, non-interference type systems, etc. These concepts undergo fast-paced development (esp. where concurrency is involved), and no general agreement exists yet in this field.

B.3 Reasons for the Action

Overcoming the fragmentation in object-oriented software verification not only benefits the field by allowing joint work on common challenges. It also benefits the software manufacturers as the users of verification technology because its usefulness is greatly enhanced if (a) industrial programming languages are fully covered and (b) the techniques and tools are integrated and use common specification languages, so that one can easily use different tools with different strengths (precision, automation, finding counter examples, construction of test cases). To provide standardised specification methodologies, languages, and interfaces for tools is a further goal of this Action.

The COST Action responds to the need for co-ordinating and reinforcing research in formal verification of object-oriented software. This will result in verification technology with greater reach and power, which can be applied to realistic, industrial software. Thus, higher-quality software can be produced while at the same time reducing the cost of software testing. The new technologies can benefit thousands of software companies in Europe.

The successful implementation of the Action will contribute to maintain Europe's leading position in the field of formal methods in software engineering with significant potential for the numerous European high-tech industries (e.g., automotive, aerospace, smart cards) that use software in safety- and security-critical areas.

B.4 Complementarity with other research programmes

The Action complements the following projects within the EU Framework Programme:

- CREDO: Modelling and Analysis of Evolutionary Structures for Distributed Services (www.cwi.nl/projects/credo)
  This project concentrates on the formal specification of dynamically reconfigurable distributed systems. While formal methods play an important role in the CREDO project, it differs from the Action in that
    - it focuses on a particular type of applications that are both dynamic and reconfigurable and the particular problems that entails;
    - it does not aim at full verification of programs; and
    - it is not concerned with the particularities of industrial object-oriented programming languages.

- MOBIUS: Mobility, Ubiquity and Security (mobius.inria.fr)
  The MOBIUS project aims to develop the technology for establishing trust and security in distributed systems, using the proof-carrying-code paradigm. It differs from the Action in that
    - it focuses on a particular type of applications that are distributed and the particular problems that entails;
    - it uses a particular methodology, namely the proof-carrying code paradigm;
    - it does not aim at full verification of programs; and
    - it is not concerned with the particularities of industrial object-oriented programming languages.

  S3MS: Security of Software and Services for Mobile Systems (www.s3ms.org)
  The objective of the S3MS project is to create a framework and a technological solution for trusted deployment and execution of communicating mobile applications in heterogeneous environments. It is concerned with formal specification of applications, but not with their formal verification (instead, the specified properties are checked at run-time).

C. OBJECTIVES AND BENEFITS

C.1 Main/primary objectives

The main objective of the Action is to develop verification technology with the reach and power to assure dependability of object-oriented programs on industrial scale.

C.2 Secondary objectives

The secondary objectives of the Action are:

- The development and standardisation of specification languages and methods for object-oriented programs that
    - cover all features of modern languages used in the software industry,
    - support modularisation of such programs and their correctness proofs.
- The standardisation of tool interfaces; and the integration of available tools into a common framework.
- To give potential users easy access to verification technology and tools, and to educate them in their use.

- The co-ordination of European research in the field of object-oriented program verification to
    - eliminate duplication of efforts,
    - achieve synergy.
- To increase market penetration of formal verification technologies.

C.3 How will the objectives be achieved?

The means to achieve the objectives include:

- regular meetings of the scientists involved,
- joint research and publications (scientific contribution to international conferences and journals),
- setting common goals and developing common benchmarks,
- systematic training of young researchers, and
- collaboration with the industry.

C.4 Benefits of the Action

The Action will facilitate exchange of knowledge and encourage new research in verification of object-oriented software in Europe. The scientific community will benefit in form of a faster progress in the field as efforts are bundled and duplication is avoided.

Software developers in the industry will have easier access to different powerful verification technologies, each of them with their own benefits and advantages, within an integrated tool set. Using these tools will result in higher-quality software.

The consumers will benefit from more dependable software available at reduced prices. Improvement in safety and security of software will contribute to the well-being of the general public in the European Union and worldwide.

The results of the Action can help in the improvement and advancement of standards for the certification of safety- and security-critical software. This, in turn, benefits the software industry and the general public that uses critical software.

C.5 Target groups/end users

The target groups of the Action and the end users of its results (in the ways outlined in the previous sections) are:

- the scientific community,
- the software industry,
- the general public that uses critical software,
- teachers and students in software engineering,
- standards bodies.

D. SCIENTIFIC PROGRAMME

D.1 Scientific focus

The Action will bring verification technology to a state where it has the reach and power to assure reliability of object-oriented programs on industrial scale.

Reach means the ability to verify real industrial programs. Such programs regularly tap the full feature set of modern object-oriented programming languages. Thus, to have real practical value and achieve a noteworthy market penetration, verification must fully cover these languages, as opposed to just certain subsets as is the case now.

Power means the ability to fully verify large-scale applications as opposed to small parts or small programs. Experience shows that the key to handling large programs is modularisation - separating the application into parts, which can be verified independently. Another aspect of power is the usability of methods and tools, which has to be significantly improved for the adoption in the industry.

The scientific program to achieve these goals comprises the following steps:

- catalogue in detail the strengths and weaknesses of different verification approaches;
- use technology transfer within the network to extend the coverage of individual approaches;
- where no further extension is possible, develop combined methods within a common platform (e.g., static analysis and theorem proving in one system);
- where no further method integration is possible, develop standard interfaces between methods and tools - make it possible to import results from one tool into another;
- measure progress on a industrial benchmarks.

The Action will carry out this work in four Working Groups:

- WG 1: Adaptable and Reusable Programs
  This WG will extend the reach of verification by addressing program features that are not yet standard in current verification approaches.

- WG 2: Modularisation and Components
  This WG will extend the power of verification by harmonising and advancing modularisation concepts.

- WG 3: Concurrency, Mobility, Distributed Systems
  This WG is concerned with both reach and power in verifying concurrent systems, but is a separate working group due to the complex and important nature of the topic.

- WG 4: Tool Integration
  This WG will develop the common platform for the new integrated generation of tools.

D.2 Scientific work plan – methods and means

The Action's scientific work plan is structured in the Working Groups as follows:

WG 1: Customisable and Reusable Programs

Current versions of modern programming languages have mechanisms for reusing programs and adapting them to particular applications (the most prominent mechanisms being aspects, generics, and reflection). The industry uses these mechanisms more and more in order to increase customisation and reuse of code, decrease costs and shorten product cycles. In this scenario, a customisable program constitutes merely a blueprint, which can be used to produce a whole series of customised software products.

As matters stand, quality assurance based on testing is harder for such customisable programs than for their conventional counterparts. A customisable program may be used to produce a large number of customised instances, all of which have to be tested. The differences between instances can be significant and spread throughout the code.

It would be beneficiary to apply logic-based techniques to verify the customisable program in its blueprint form once and for all, without considering different instances. Current verification technology does not cover customisable programs adequately though. WG 1 will extend the existing object-oriented specification and verification techniques to cover the new customisation features.

The working group will:

- identify common concepts of customisable programs across different programming languages;

- extend specification languages with means to specify customisable programs;

- extend verification calculi to handle customisation concepts;

- implement the extended calculi in verification systems;

- evaluate the extended languages and calculi using benchmarks.

The activity of this Working Group will be closely co-ordinated with WG 2 on the issues of modularisation (see below).

WG 2: Modularisation and Components

Verification technology can only be successful in an industrial environment if it scales to treat large programs. This requires modularisation of programs and, thus, corresponding verification tasks and proofs. WG 2 will concentrate on specification techniques that allow effectively modularising programs and specifying modules and interfaces. Proof methods will be developed making use of these. The group will also agree on modularisation mechanisms for inclusion in existing specification languages.

The need for program modularisation has been understood for a long time. This need became even more pressing with the emergence of object-oriented programming. With language features like polymorphy and inheritance, reference aliasing, and shared-memory multi-threading, local programmer decisions have non-local impacts and make reasoning about large programs difficult.

There is a broad consensus that the modularisation problem has not been satisfactorily solved with the languages used in practice today, such as Java, C#, and C++. Thus, development of modularisation concepts has become a task for verification experts. Several partial solutions have been proposed for this problem, while no general agreement yet exists.

Every modularisation concept has to answer three questions: What are suitable modules in programs? How can the module boundaries be enforced? How can the module interfaces and behaviour be specified? There is also an important trade off to make. Effective modularisation requires putting restrictions on what programs can do, limiting undesired interference between parts. This complicates programming. Less restrictions, on the other hand, offer a higher usability, but at the cost of more difficult verification and a higher specification overhead.

The research will evaluate and reconcile different modularisation techniques, such as ownership type systems (universes, box models, etc.), abstract descriptions of interfaces (model methods and fields), non-interference analyses, and refinement relations.

The Working Group will:

- identify common modularisation concepts that gives a good balance between power and usability;
- extend established specification languages with module descriptions;
- develop powerful means for enforcing module boundaries (e.g., ownership checkers);
- extend verification calculi to make use of modules, allowing verification to scale;
- implement the concepts within existing tools;
- evaluate the concepts and extended calculi using benchmarks.

This WG will interface with WG1, since certain aspects of customisable problem (generics) can ease modularisation, while others (reflection) have a negative impact. Also, cooperation with WG 3 on matters of concurrency will take place.

WG 3: Concurrency, Mobility, Distributed Systems

Concurrent programs are of an increasing importance for modern computing, with its proliferation of multi-core, distributed, and mobile systems. Their verification, however, is an area where formal methods are most lacking and need extension and adaptation to the requirements of the software industry. This will be the goal of WG 3.

Power: Verification of concurrent and distributed systems draws its power from identifying parts of code that can be considered sequentially. This strategy is similar to the one described under modularisation. The problem has been tackled by different groups, but significant advances can still be achieved by bundling efforts.

Reach: Lately the popularity of performance-optimised, highly-concurrent algorithms and lock-free/wait-free data structures has enjoyed immense growth. These algorithms are notoriously difficult to design and implement correctly. Verification of their implementations is an important goal, but current methods of treating concurrency in verification are geared towards programs with benign serialisability properties. They do not scale well to highly-concurrent algorithms. WG 3 will develop solutions to this problem.

This Working Group will:

- develop an efficient modularisation concept for concurrent programs (in co-ordination with WG2);
- based on this concept develop ways in which different verification techniques (static analysis, sequential deductive verification, model checking) can be integrated;
- develop and implement methods for verification of highly-concurrent programs;
- evaluate the new methods using benchmarks.

WG 4: Tool Integration

To give users easy access to the tools of the research groups participating in the Action, WG 4 will standardise tool interfaces and develop a framework for integrating the different tools (and their benefits).

This Working Group will:

- standardise specification techniques/languages in the verification community (beyond particular aspects treated by other WGs);
- develop a framework/platform for integration of tools. This includes a common infrastructure (intermediate representations and data structures) that allows information sharing beyond simple input/output. The common framework will yield synergies from combining different types of program analyses and increase verification power.
- develop new, integrated software analysis tools based on verification technology, but not limited to functional verification. The range of tools will, for example, include execution visualisation, bug finding, bug localisation and presentation, and white-box test generation. This will increase verification reach;
- promote the use and evaluation of the integrated verification tools.

Realistic Challenges

The industrial partners of the Action will contribute realistic challenge programs of various sizes and degrees of difficulty. They will help the academic partners to benchmark technologies and the performance of tools, and to understand what kind of expectations the industry has regarding the application of formal verification.

E. ORGANISATION

E.1 Coordination and organisation

The Action will be co-ordinated by the Management Committee (MC) that is presided by a Chair and Vice-Chair. Scientific work will be carried out in four Working Groups (WGs).

The Management Committee

The main responsibilities of the MC that co-ordinates the Action are:

- appointment of Chair, Vice-Chair(s) and the WG Co-ordinators;
- planning and co-ordination of the different meetings: MC meetings, scientific meetings as well as workshops and conferences;
- assessment of the different activities (Short-Term Scientific Missions, publications, training schools, etc.) in order to meet the objectives of the Action;
- report of the progress made by different WGs to meet their respective objectives in the framework of the Action;
- promotion of collaboration and of exchange of knowledge (and data) between the partners from different WGs;
- promotion and approval of Short-Term Scientific Missions;

- creation and regular updating of a website in order to enhance communication between partners and to disseminate the results generated in the different WGs;
- co-ordination and facilitation of all efforts that can lead to the preparation of research project proposals related to the topics of the Action (the aim is to establish at least one EU Seventh framework project);
- preparation of annual reports.

Meetings of the MC will take place twice a year linked with the WG meetings. This will insure efficient co-ordination of the activities and managing of activities and critical points of the programme.

Working Groups

Each of the WGs will be managed by a WG Co-ordinator. These WG Co-ordinators will have as main tasks:

- participate in the meetings of the MC;
- plan the appropriate scientific meetings;
- co-ordinate the activities within their WG in order to meet the objectives that are defined in the scientific programme;
- promote the set-up of joint research (e.g., making use of Short-Term Scientific Missions);
- promote the writing of common publications;
- report the WG progress to the Action chair and MC.

WG Meetings

The WGs will meet twice a year in changing host countries. The meetings will be accompanied by thematic workshops.

These frequent gatherings are planned in order to have an optimal exchange of ideas. It is planned to hold two to three days meetings for each WG. The first one to two days would be devoted to specific WG activities. This would allow the exchange of information and ideas, encourage the collaboration between scientists, stimulate the planning of joint publications and will address WG specific topics. The last day of the meeting would be combined with the other WGs. This would greatly enhance integration of activities from the different fields, and promote interface between WGs.

Action Conference

In years two and four of the Action, a conference will be organised, to which also other academics and users from the industry will be invited, in order to facilitate a broad exchange and dissemination of ideas.

Training Schools

In years one and three of the Action (i.e., alternating with the conferences), a training school for young researchers will take place. The schools will build researchers' skills and promote excellent practices. They also help overcome the fragmentation of the field, as young researches learn about

different techniques and methodologies for formal program verification.

A practical course will be organised as an intensive one-week training available to anyone, in particular software practitioners, from participating COST countries who wants to acquire hands-on experience of modern verification techniques.

Short-Term Scientific Missions

Short-Term Scientific Missions will enhance exchange of knowledge, technology transfer, and training in new techniques. Moreover they will strengthen the collaborations between partners, promote joint research and development of a long-lasting European network. In promoting STSMs the MC will particularly favour the mobility and training of young researchers.

E.2 Working Groups

The Action has four Working Groups, which are listed in Section D.1. Their work plans are described in Section D.2. Organisational details are outlined in Section E.1 above.

E.3 Liaison and interaction with other research programmes

The Action will liaise with the EU FP projects mentioned in Section B.4 and other international projects, including the Grand Challenge initiative and the German Verisoft project. This interaction will be facilitated by participants of the Action who also take part in the other projects. Furthermore, key representatives of other research programmes will be invited to participate in selected activities of the Action (meetings, conferences, etc.).

E.4 Gender balance and involvement of early-stage researchers

This COST Action will respect an appropriate gender balance in all its activities and the Management Committee will place this as a standard item on all its MC agendas. The Action will also be committed to considerately involve early-stage researchers. This item will also be placed as a standard item on all MC agendas.

The MC will explicitly encourage participation of young researchers in all activities of the Action. The training schools for young researchers (taking place in years 1 and 3) will build up expertise and help overcome the fragmentation of the field, as young researches will learn early on about different techniques and methodologies for formal program verification. The short-term exchange (via STSMs) of young researchers between the participating groups will be a central pillar of the Action, as it permits young researchers to develop lasting contacts.

The participants in the Action will be encouraged to promote the involvement of female professionals in the different activities, and to adhere to gender equality when selecting and/or appointing new recruits. The gender balance will be overseen by the Management Committee.

F. TIMETABLE

- The duration of the Action is 4 years.

- The Kick-off Meeting will start the Action, and during this meeting the WG Co-ordinators will be appointed.
- The Action website will be created soon after the Kick-off Meeting and will be updated continuously.
- Every year a meeting of the Management Committee and the Working Groups will take place.
- In years one and three of the Action, a training school for young researchers will take place.
- In years two and four of the Action, a conference will be organised. The conference in year four will be the final event of the Action.
- Short-term Scientific Missions can be requested any time after the first WG Meeting.
- Thematic seminars/workshops will be organised with an open schedule and according to the needs of the Working Groups.

The timetable of the Action is summarised in the following diagram:

	Year1		Year 2		Year 3		Year 4	
Co-ordination	X	X	X	X	X	X	X	X
Kick-off meeting	X							
Homepage	X	X	X	X	X	X	X	X
Reporting		X		X		X		X
MC meeting	X		X		X			X
WG meetings	X		X		X		X	
Training school		X				X		
Conference				X				X

G. ECONOMIC DIMENSION

The following COST countries have actively participated in the preparation of the Action or otherwise indicated their interest: BE, EE, FR, DE, IE, IL, NL, PL, SE, UK. On the basis of national estimates, the economic dimension of the activities to be carried out under the Action has been estimated at 10 Million € for the total duration of the Action. This estimate is valid under the assumption that all the countries mentioned above but no other countries will participate in the Action. Any departure from this will change the total cost accordingly.

H. DISSEMINATION PLAN

H.1 Who?

Target audiences for the dissemination of the results of the Action are:

- the scientific community,
- software industry (both in Europe and world-wide),
- standards bodies,
- students and teachers in the field of computer science and software engineering,
- the general public.

H.2 What?

The results of the Action will be disseminated in a number of ways, including:

- press releases;
- refereed scientific publications;
- technical documents and guidelines;
- the Action website;
- meetings, workshops and conferences;
- education at university level.

H.3 How?

Press releases

The first press release will be launched in all participating countries on the occasion of the project kick-off meeting. Further press releases will inform the general public of important results of the Action.

Publications

Scientific results of the Action will be disseminated through refereed scientific journals and conference proceedings. The MC will promote co-publications as much as possible.

Technical guidelines focussing on one or more technical aspects related to formal verification, mainly those linked to the activities of WG 4 (Tool Integration) will be produced.

Website

A public website will provide information to the international scientific community, to the industry, and to the general public. It will also facilitate communication flow between the partners of this project. This website will be maintained by the MC. The website will combine a public part accessible to everybody with an access-restricted intranet for the participants.

The website will contain, among other things:

- general information about COST and this Action, including its activities (meetings, etc.) and contact information of Action participants;
- publications of the participants relevant to the Action;
- information for the public and the target groups in the software industry;
- on-line courses, proceedings, slides and posters from training schools and WG meetings;
- STSM calls and reports;
- teaching materials (slides, course notes, etc.);
- software developed by the participants, in particular their program verification systems;
- links to websites related to verification;
- job announcements.

Workshops

The consortium will organise workshops to inform interested scientists, practitioners from the industry, standards bodies and policy makers about the results of the project and about new technologies developed throughout the project. These workshops will provide hands-on practical training as well as theoretical information.

STSM

Short-Term Scientific Missions (STSMs) will facilitate scientific networking, technology transfer within the Action, and training in new techniques.

International conferences

Knowledge and data resulting from the COST Action activities will be integrated and presented at international conferences. This will promote the European know-how and increase the international collaboration.

Teaching activities

Teaching activities in universities at undergraduate and post-graduate level will take advantage of the knowledge and experience acquired during this COST Action. Young scientists and engineers will be informed and trained in the latest developments in verification of real-world software.