Table of contents

1. PROJECT SUMMARY **4**

2. PROJECT OBJECTIVES **6**

3. PARTICIPANT LIST **8**

4. CONTRIBUTION TO PROGRAMME/KEY ACTION OBJECTIVES **9**

5. INNOVATION **10**

6. COMMUNITY ADDED VALUE AND CONTRIBUTION TO EU POLICIES **12**

7. CONTRIBUTION TO COMMUNITY SOCIAL OBJECTIVES. **13**

8. ECONOMIC DEVELOPMENT AND S&T PROSPECTS **14**

9. WORKPLAN **16**

- 9.1 General description: Overall project structure 16
- 9.2 Workpackage list 17
- 9.3 Workpackage descriptions 18
- 9.4 Deliverables list 32
- 9.5 Project planning and time table (GANTT Chart) 34
- 9.6 Graphical presentation of project components 35

9.7 PROJECT MANAGEMENT **36**

10. CLUSTERING **38**

11. OTHER CONTRACTUAL CONDITIONS **38**

REFERENCES **39**

APPENDIX A - CONSORTIUM DESCRIPTION **41**

- Institut National de Recherche en Informatique et Automatique (F) 42
- France Télécom R&D (F) 44
- University of Firenze (I) 46
- University of Sussex (UK) 48
- University of Lisboa (P) 50

APPENDIX B - CONTRACT PREPARATION FORMS **51**

APPENDIX C – GANTT CHART OF THE PROJECT .... ERREUR ! SIGNET NON DÉFINI.

1. Project summary

Current middleware and programming language technologies are inadequate to meet the challenges posed by a global computing environment. In particular, they tend to support only a limited range of interactions, have a limited view of components and objects, fail to properly and uniformly support properties such as mobility, predictability, security, fault-tolerance, and they are not amenable to rigorous investigation for verification, validation and test purposes.

The Mikado project intends to overcome these limitations by defining and prototyping new formal models for both the specification and programming of highly distributed and mobile systems, and to develop specification and analysis techniques which can be used to build safer and trustworthy systems, to demonstrate their conformance to specifications and to analyse their behaviour.

Shared Cost RTD CPF Form – Form A2

EUROPEAN COMMISSION RESEARCH DIRECTORATES GENERAL SHARED COST RTD CPF FORMS	EN C 1 FP5RTD FOR COMMISSION USE ONLY

Project Acronym [2]	MIKADO	**Proposal No** [3]	IST-2001-32222

A2. Project Summary [20]

Objectives (maximum 1000 characters)

The goal of the Mikado project is to study new formal programming models, based upon the notion of domain as a computing concept, which support reliable, distributed, mobile computation, and provide the mathematical basis for a secure standard for distributed computing in open systems. Specifically, Mikado intends :
- To develop new formal models for both the specification and programming of large-scale, highly distributed and mobile systems.
- To develop new programming language features supporting such models, and to study their combination with functional and object-oriented programming.
- To develop specification and analysis techniques which can be used to build safer and trustworthy systems, to demonstrate their conformance to specifications, and to analyse their behaviour.
- To prototype new virtual machine technologies which can be used to implement in a “provably correct” way such models and languages.

Description of the work (maximum 2000 characters)

The project is organised around three technical work-packages (WP1-WP3) and one organisational work-package (WP4):
- **WP1: Programming Models**. WP1 is concerned with the definition of a parametric programming model for global computing, based on the notion of domain. Together with the study of different domain-based models for distributed programming, this will provide the basis for the rest of the theoretical work taking place in WP2 and for the development work taking place in WP3.
- **WP2: Specification and Analysis**. WP2 is concerned with the definition of Specification and Analysis technologies for the project's programming models. This will range from the development of type systems and static analysis techniques for expressing constraints on concurrency, mobility and resource access for the underlying execution model, to providing proof technologies for assuring that mobile code, and more generally distributed systems, conform to predefined behavioural specifications. The latter will require the definition of novel co-inductive techniques for comparing the distributed behaviour of systems and the elaboration of new specification logics for expressing interesting partial views of systems and programming paradigms.
- **WP3: Virtual Machine Technology and Language Support**. WP3 is concerned with the embodiment of the Mikado programming models developed in WP1 and WP2 in concrete programming technologies. Work in WP3 will be concerned with the development of several prototypes, including: virtual machine technology to support WP1's core programming model together with WP2 typing schemes ; languages features and language extensions supporting WP1's model and WP2's type systems.
- WP4: Project Co-ordination and Dissemination.

Milestones and expected results (maximum 500 characters)

Year 1: Analysis of programming requirements for global computing. Analysis of the state of the art in formal models, type systems, and execution structures for distributed mobile computing.

Year 2: Parametric programming model and type systems, Co-inductive proof techniques, Analysis of component properties, Core distributed virtual machine framework.

Year 3: Specification logics, Domain-specific patterns for distributed virtual machines, Programming languages for global computing.

Year 4: Dissemination and participation to final Global Computing event.

2. Project objectives

The overall goal of the Mikado project is to study new formal programming models, based upon the notion of domain as a computing concept, which support reliable, highly distributed and mobile computation, and provide the mathematical basis for a secure standard for distributed computing in open systems.

The term "global computing" is used to describe a projected scenario in which processors will be everywhere, e.g. in household devices, in cars, in clothing etc, and in which networks will be heterogeneous and highly diverse in their capabilities, e.g. high-speed networks, ad-hoc wireless networks etc. They will be able to communicate and also sense and interact with their environment. The result is a "*massive networked infrastructure composed of highly diverse interconnected objects that should support the design and use of systems with a predictable and desirable behaviour*"[1]. Such systems are also typically *autonomous*, inherently *mobile*, have configurations that *vary* over time and operate on the basis of *incomplete* information. Furthermore, they are likely to consist of a mixture of personal and more public devices. The challenge is then to "*define and exploit dynamically configured systems of mobile entities that interact in novel ways with their environment to achieve or control their computational tasks*"[1].

Current middleware and programming language technologies are inadequate to meet the challenges posed by such an environment. In particular, they tend to support a limited range of interactions, have a limited view of components and objects, fail to properly and uniformly support properties such as mobility, predictability, security, fault-tolerance, and are not amenable to rigorous investigation for verification, validation and test purposes. In a global computing environment, characterised by its openness and its ubiquity, these limitations become real design and construction bottlenecks. In order to overcome these different limitations, the Mikado project intends to develop new formal models for global computing. Models with a formal (i.e. mathematical) semantics, should provide a sound basis for constructing global computing systems which are "sound by construction" and which behave in a predictable and analysable manner. Indeed, providing quality of service guarantees in such an open and large-scale context, where "quality of service" refers to broad non-functional properties such as security, responsiveness, fault-tolerance, real-time, availability, etc, requires sophisticated and rigorous means of analysis and verification, which only formal models can provide.

Specifically, the project has the following objectives :

- To develop new formal models for both the specification and programming of large-scale, highly distributed and mobile systems;
- To investigate new programming language features supporting such models, and their possible combination with other programming paradigms such as functional and object-oriented programming;
- To develop specification and analysis techniques which can be used to build safer and trustworthy systems, to demonstrate their conformance to specifications, and to analyse their behaviour;
- To investigate and prototype virtual machine technologies which can be used to implement in a "provably correct" way such models and languages.

A major modelling issue that the Mikado project will investigate is the use of "domains" as a uniform means of describing the different forms of partitions, boundaries and structures that become necessary in a global computing environment. These domains could take many forms to account for the various administrative and technical boundaries required for the provision of security, mobility management,

[1] See http://www.cordis.lu/ist/fetgc.htm#vision.

accountability, fault management, etc. Providing means to handle different forms of domains uniformly at the programming language level will be one of the key contributions of Mikado.

In addition to the issue of modelling, of great importance is the provision of means to program the different forms of interaction, structuring and control necessary in highly mobile, open, large-scale computations, e.g. going beyond the traditional but limited point-to-point process interactions to provide for different forms of multi-party interaction, process superposition and system composition.

Having provided powerful programming constructs to express sophisticated distributed and mobile computations, providing ways to statically constrain (e.g. by means of type systems) these computations so as to meet certain safety conditions is absolutely essential. Example properties which will be considered include absence of communication errors, absence of security errors (e.g. when controlling access to resources or securing information flows) and absence of resource-availability errors.

Some of the more specific issues that the Mikado project will investigate include:

- New mobile process calculi (e.g. in the tradition of the π-calculus and its variants) that can embody the different programming constructs necessary for global computing in a primitive form, together with their formal semantic basis (i.e. different mathematical models for the operational or denotational semantics of these calculi).
- Issues associated with the combination of new "global computing" programming constructs with more traditional programming paradigms such as e.g. functional, imperative and object-oriented programming.
- New type systems and static analysis techniques (e.g. in the tradition of type systems and control–flow analysis for the π-calculus and its variants) that can help capture and enforce interesting safety properties of large-scale distributed computations.
- New operational structures and models for distributed virtual machines that can support the project's programming languages and process calculi (e.g. in the tradition of virtual machine support for functional and parallel programming languages).
- Semantic equivalence notions and associated proof techniques required for proofs of program equivalence and proofs of correction.
- Specification logics for the expression of behavioural properties of components, processes or domains in a global computing environment.

Additional issues having to do with the open and heterogeneous character of a global computing environment and which Mikado will consider include :

- Issues associated with the introduction of polymorphism and dynamicity in type systems, two features which can be necessary in an open environment where typing information could be partial, inaccurate, missing or erroneous.
- Issues associated with the co-existence of different programming languages and models in a large scale global computing environment, such as, e.g. the combination of different interaction models or of different type systems.

The success of the project will be measured by its level (quality, number) of publication in high-quality, international scientific journals and conferences, which will provide an indication of the relevance and quality of the theoretical basis developed by the project, and by its ability to translate these theoretical results into working software code (virtual machine and language technologies).

3. Participant list

List of Participants

Partic. Role*	**Partic. no.**	**Participant name**	**Participant short name**	**Country**	**Date enter project****	**Date exit project****
C	1	INRIA	INRIA	F	Start of project	End of Project
P	2	France Télécom R&D	FTRD	F	Start of project	End of Project
P	3	University of Florence	DSI-UF	I	Start of project	End of Project
P	4	University of Sussex	UOS	UK	Start of project	End of Project
P	5	University of Lisbon	FFCUL	P	Start of project	End of Project

4. Contribution to programme/key action objectives

The subject of the Mikado project is directly relevant to the objectives of the IST programme for constructing a global computing space that is universally and seamlessly accessible to all. The contribution of Mikado to this objective is the provision of essential technology for building dependable infrastructure and applications. In particular, Mikado models and programming technology directly target the construction of "efficient computing infrastructure", which, "together with advanced mobile and networked embedded systems", "enable anywhere/any time access to services". The Mikado project responds directly to Action line VI.2.2 "Global computing : co-operation of autonomous and mobile entities in dynamic environments".

The proposed research in Mikado addresses foundational aspects of "global computing" programming, i.e., theoretical models and calculi for specifying, programming, and reasoning about global computing systems. The work is long-term, and can lead to major new insights into the fundamentals of interaction and control in large-scale, mobile, distributed computing. In particular, we expect Mikado to provide new programming constructs that allow system designers and programmers to deal in a uniform way with the various forms of "domains" that will be so pervasive in a global computing environment. The research is also supported by more practical work dealing with the definition and implementation of new programming languages based on these models and calculi. As such, the project directly addresses the following issues, raised in the Global Computing Initiative:

- Models of computation and models of communication / interaction / co-ordination : Mikado will consider programming models focusing on the temporal, spatial and logical structuring of large-scale, mobile, distributed computations, including not only different forms of interaction and co-ordination but also different forms of superposition and composition for the distributed control of such computations. Mikado's core programming model will draw from the expertise acquired on distributed process calculi but its insight of domains as first-class entities will constitute a major innovation (and a major modelling challenge) compared to the current state-of-the-art.
- Programming : Mikado will investigate programming languages based on the above process calculi, as well as their combination with well-known programming paradigms.
- Security and resources : basic domain constructs expected in the Mikado model should provide the means to specify and program systems which are partitioned in numerous ways, notably including security domains (as enforced e.g. by sub-network firewalls) and resource domains (as caused e.g. by a spatial distribution of resources). Type systems investigated by Mikado should provide the means to enforce statically-verifiable safety constraints on computations such as e.g. absence of interaction errors, of security errors (e.g. enforcing specified security levels) and of resource-availability errors.

The focus of the Mikado project on a programming model for global computing is perfectly on line with the focus of the global computing action line and its three key aspects:

- The Mikado programming model should constitute a key element for the design of systems of autonomous entities, with highly dynamic and decentralised behaviour.
- Mikado's emphasis on formal models and semantics is crucial to enable the formal analysis of, and reasoning about the behaviour of such systems. Mikado's work on co-inductive proof techniques and specification logics for global distributed computing systems should prove invaluable tools in this respect.
- The work on static analysis and type systems in Mikado will answer directly to the challenge of avoiding and detecting undesirable behaviour in such systems. Mikado's core programming model will provide basic primitives for specifying and programming distributed control of such systems, including resources organised in complex webs of security and dependability domains.

5. Innovation

The Mikado project intends to depart radically from current distributed object-based and component-based programming models, such as those based on the standard distributed system platforms (e.g. OMG CORBA, Sun Java RMI/EJB/Jini, Microsoft COM/Universal Plug&Play/.Net), those exhibited by recent mobile agents platforms (e.g. Voyager, Aglets, Grasshopper), or those exhibited in experimental platforms and languages such as Network Objects [6], Orca [4], Kali Scheme [12], Facile [19], Obliq [9], etc. These technologies improve on the traditional client-server programming model, e.g. introducing ideas of shared objects, multi-faceted components, migrating agents, or mobile code, but they fail to provide a uniform, formal model of distributed and mobile computing in a large-scale, open, highly-partitioned, global computing environment. In particular, the core of these technologies is usually based on simple assumptions (such as, for example, a simple model of connectivity à la TCP) which are bound to be invalidated in a global computing environment with a wide diversity of operating conditions, complex communication and resource topologies, very different application requirements, highly dynamic and mobile configurations. Standard distributed platform designers are aware of this situation and have responded with a flurry of specific additions, extensions and alterations to mainstream specifications such as OMG CORBA (e.g. Minimal CORBA, Real-time CORBA, CORBA Security, Fault-tolerant CORBA, CORBA Component Model, Mobile CORBA, etc) and Sun Java (Real-time Java, Personal Java, Java Embedded, Jini, Enterprise Java Beans, etc). The result is an unpalatable array of specifications with diverging implicit programming models, with hard-to-analyse, informal semantics.

Current research on infrastructure and middleware for distributed systems, taking into account new concerns such as mobility and adaptability has resulted in the introduction of various distributed programming models, including e.g. support for mobility [21,23], support for large-scale distribution and replication [31], support for real-time and multimedia constraints [29], support for event-based computations [3]. Since most of these works suffer from the same problem of diverging models, a recent body of work introduces some form of structural and behavioural reflection [7,8,20,24] in middleware systems to try and provide a more principled approach to the support of distributed and mobile computing aspects. While it is too early to see whether such reflective approaches to distributed middleware construction might work, it is clear that the programming models provided by these experimental platforms remain highly informal and implicit (i.e. they are more the by-products of a platform features than explicit constructions). As a result, just as for mainstream industrial platforms, distributed computations executing on such systems are very hard, if not impossible, to analyse and reason about. Without a formal basis, there is no hope to obtain the level of analysability required for designing and constructing open, dependable, large-scale, distributed mobile systems.

Turning to the area of formal models for concurrent and object-oriented programming, several formalisms and languages have been proposed and studied, in particular from the point of view of type systems and proof techniques. A large number of proposals are in fact variants of the π-calculus [27], or some forms of object-based or actor-based calculi (see e.g. [1,32]). More recently, several formal models for distributed and mobile computing have been proposed, including, for example, the Join calculus [16], the π_{11}-calculus [2], the Ambient calculus [10], the Safe Ambient calculus [25], the Seal calculus [37], the Dπ-calculus [22], DiTyCO [26], Nomadic Pict [28], KLAIM[14], Oz [30] (see also [11] for a survey of process calculi with localities). Also, various extensions of the Actor model [18] have recently been proposed which aim to adapt the actor model of computation to large-scale distributed and mobile computing. Very often, these models centre around notions of localities or sites, as championed by the ambient calculus, for spatially partitioning computations and for handling asynchronous communication, failure, mobility, and security aspects. None of these models, however, provides a uniform and consistent set of concepts and primitives for dealing with the different forms of components and domains that typically appear in a large-scale, open distributed system. In particular, for dealing with various distributed computation aspects such as security, mobility, fault detection and management, one can find a vast array of proposed primitives and type systems, the direct comparison and assessment of which remain difficult.

Compared to the current state of the art, the key innovative elements of Mikado will comprise:

- A parametric model of migration and mobility that takes into account key features of computation in a global computing setting: distribution and mobility, multiple forms of components and of domains, multiple forms of interaction between components. The notion of domain is a key insight for developing the Mikado programming model. Briefly, a global computing environment is probably best understood as a collection of regions, or locations, separated by barriers and boundaries of many different sorts (see e.g. [38] for a discussion of locations, barriers and boundaries in wide area computing). Such regions and their associated barriers, which we call domains in Mikado, may take many different forms and correspond to a wide variety of aspects. For instance, domains may correspond to security and privacy regions, to failure confinement zones, to control and management scopes, to communication cells, etc. The key assumption in Mikado is that, despite the diversity of domains in any large scale distributed system, the concept of domain can be leveraged into a few basic primitives for structuring and controlling distributed computations. This basic insight is similar to that which led to the definition of the Ambient calculus, of the Dπ-calculus or of the distributed Join calculus with their notions of localities. Mikado's approach of domains as first-class entities, however, constitutes a rather radical generalisation of these earlier approaches, with the goal to allow, using the Mikado programming model, for the definition and construction of arbitrary forms and behaviors of domains (e.g. domains endowed with access control features, specific behaviors in presence of faults and failures, etc)[2].
- Novel type systems and static analysis techniques that extend recent work on type systems for object-oriented languages and distributed process calculi to take into account constraints such as the prevention of interaction and deadlock errors, of resource-availability errors, or of security errors. Particular consideration will be given to the domain-based, type-theoretic handling of security and dependability issues, including e.g. a type-theoretic formulation of "proof-carrying code" and a type-theoretic characterisation of failure detection.
- Sound semantic foundations for component and systems equivalence in a global computing setting, together with associated logics for the specification of behavioural constraints and properties, both essential for the development of verification tools and techniques and, ultimately, of provably correct global computing systems.
- Novel language constructs and supporting distributed virtual machine technology, including the combination of Mikado's domain-based constructs with object-oriented and functional programming languages, and provably correct virtual machine prototypes for Mikado's programming model. Obtaining proof of correctness for distributed virtual machines will be a substantial achievement of the Mikado project, since such proofs have very rarely been attempted and only in more limited settings (see e.g. [17,33]).

[2] By contrast, current distributed process calculi take particular instances of domains as primitive and do not allow the definition of arbitrary forms of domains.

6. Community added value and contribution to EU policies

Over the last decade, the increasing availability and usage of new computing devices (embedded systems, smart cards, mobile phones, personal digital assistants, laptops, ...) and the rapid penetration of the Web have created new ways of working and living together in society. For example, new work and business practices (e-commerce, telework, telemedicine, ...) have emerged, and new services for individuals (online shopping, chat services, onboard road navigation in cars, online filing of tax returns, ...) are starting to be widely available. These new services and new ways of human interaction are beginning to permeate all levels of European society, and it is common expectation that they should be highly dependable, adaptable, mobile and universally accessible. With the advent of global computing, computing technology will become even more pervasive, will reach new levels of automation and autonomous behaviour, while society at large will become even more dependent on it. In such a context, high dependability, adaptability, mobility and universal access are exacerbated as critical properties.

However, today's software technologies for distributed and mobile computing are inadequate to fulfil these expectations. In the view of the Mikado project, a major problem with existing software technologies is that they lack adequate programming models with clearly-defined formal semantics. We believe that such models are needed to enable software infrastructures to provide true reliability, mobility and dynamic adaptability of applications and services.

In the pursuit of this goal, the project will contribute at a European level in two ways:

- By combining the strengths of two vibrant European research communities: formal models of concurrency, mobility and object-oriented programming (process calculi, type theory, ...), and distributed execution platforms (distributed operating systems, middleware architectures, ...). These two research communities are very active at the European level, and the same critical mass of expertise does not exist at national levels. Moreover, the expertise of these two European communities is recognised on an international scale.
- By creating opportunities for innovation and standardisation. Over the last ten years, an important number of developments have occurred in the two fields above, and the time is ripe to explore common ground between them to define provably correct distributed infrastructures. The mix of technology development concerns with formal foundations presents a unique combination of subjects that could create important opportunities for Europe to lead next generation Internet technologies. In particular, the projects results could be directly exploited to influence future global computing standards, either through contributions to formal or informal standard bodies (e.g. OMG, IETF), or through the development of open source global computing infrastructures (see section 8 on this last item).

7. Contribution to Community social objectives.

With its formal approach, the project is ideally positioned to lay down the foundation of future dependable and secure distributed computing systems. If the project is successful, its programming model and virtual machine technology can serve as a basis for the construction of "correct-by-construction" distributed and mobile systems, dramatically increasing the prospects for application and systems designers and developers to be able to carry out formal analysis, verification and validation of distributed and mobile application and systems. This would directly benefit the construction of critical large scale distributed systems, in application areas such as telecommunications, transport systems, electric power supply, air traffic control, industrial process control, etc. With the increased dependence of economic life on several such infrastructures, the project can have a direct influence on making these infrastructures more dependable and robust, notably by making them amenable to formal analysis and verification, thereby constituting an important contribution to general public health and safety.

Furthermore, the prospect of new programming models for distributed and mobile computation would create a unique opportunity for Europe to take the lead in middleware and Internet technology design and construction, an important and thriving area of software development with several important market niches already appearing (e.g., application servers, embedded servers, net appliances), areas where Europe currently has a very weak industrial position. Should the project be successful, technical conditions will be met for enormous development opportunities in Europe in this area, if only in critical systems markets, where the ability to build on a uniform and sound middleware and programming technology would constitute a welcome departure from today's state of the art and practices, which are dominated by costly proprietary and ad-hoc approaches to distributed systems development.

Finally, in as much as the project can provide the basic technology for future distributed and mobile systems, the project should be seen as an enabler for future innovative services, in numerous application areas, from telecommunication management to industrial process control, including electronic commerce and information management systems. The ability to handle highly mobile and reconfigurable systems in a correct and easy way, for instance, should immediately bring about new applications and services, and serve as a key enabler to global computing.

8. Economic development and S&T prospects

The economic prospects for infrastructure technology (operating systems, middleware, programming languages and virtual machines) for global computing are enormous, as can be extrapolated from the market perspectives for Java, CORBA and .Net technologies such as electronic commerce, including customer to business and business to business workflows. Beyond that, whole new markets should be available under the general heading of "ambient intelligence", which will require the safe and secure control of a multitude of embedded intelligent sensors, effectors, communicators, and information processors, be they highly mobile (e.g. handheld or embedded in clothes and other mobile objects such as cars, PDAs, phones, etc) or less so (e.g. integrated in house appliances, walls, etc).

Mikado is uniquely positioned to bring key technologies (programming languages and virtual machine technology) in this area. Considering the exploratory nature of the work undertaken in the Mikado project, it is likely that results from the project, in particular software prototypes (language support and virtual machine technology), will need to be further matured before exploitation in contexts requiring industrial-strength tools. However, we can already envisage several plans for exploiting and harnessing the potential of the project's results.

Dissemination of the project's scientific (programming model, type systems, etc.) and technological (virtual machine and language technologies) results will be accomplished through the usual means of publication in international conferences and journals, by presentations at various scientific and technical meetings and R&D institutions, and by paper and software postings on various Web sites. The project partners have a long tradition of scientific excellence in their fields and a glimpse at their recent publication records provides a good sample of the best conferences and journals that will be targeted during the course of the project.

We also have several more specific plans for disseminating the results as outlined below.

The project will establish an open workshop series focused on programming models and programming support for global computing. This will be done in conjunction with other projects in the Global Computing Initiative, and will follow the model of the highly successful series of CONCUR workshops and conferences (which some partners in Mikado have co-founded).

As a rule, the different project prototypes and software technologies developed as part of the project will be made widely available as open source software through the World-Wide Web. The project will establish a dedicated Web site, where the project's major publications and publicly available results (both theoretical results and open source software prototypes) will be posted.

For more mature prototypes developed by the project, consideration will be given for their inclusion in the code base of the ObjectWeb Consortium (http://www.objectweb.org). ObjectWeb is an open source initiative launched in late 1999 by INRIA, France Telecom R&D, and Bull/Evidian to build a comprehensive set of middleware components, all available under open source licences (e.g. LGPL or MPL licenses). Current ObjectWeb components include the Jonathan Object Request Broker, initially developed under the auspices of the ACTS ReTINA project, the Jonas Enterprise Java Beans server, whose development was partially funded by the ACTS ACTrans project, and the Joram message-oriented middleware, whose development was partially funded by Esprit project C3DS. ObjectWeb has attracted the attention of several young companies, including French start-ups Kelua, Libelis and Experlog, which have based whole or part of their business model on the exploitation of the ObjectWeb code base (typically by providing services or complementary software), and of Lutris, a US-based company which exploits the ObectWeb code base in its open source application server, called Enhydra (http://www.enhydra.org). In particular, the transaction management part of Enhydra is entirely based on Jonas.INRIA, France Telecom R&D and Evidian are currently transforming ObjectWeb into an international consortium with the goal to make ObjectWeb one of the prime reference sites concerning open source middleware.

The current ObjectWeb code base is written in Java and therefore suffers from the limitations intrinsic to the Java approach to distributed and mobile computing. Mikado's results could, in this context, largely complements the current code base or provide a path for its evolution towards more dependable middleware components. In particular, language experiments and virtual machine technology developed in Mikado could provide a substitute or an evolution of the current Java language and virtual machine technology, respectively. This could be achieved either through an hybridation of the Java language, combining domain-based primitives and type systems from the Mikado programming model with Java, or through the (less likely) displacement of the Java basis in favor of Mikado language and virtual machine technology. In either case, exploiting results from Mikado will require commensurate efforts to influence relevant standards-setting bodies in this area. The ObjectWeb Consortium, which is currently building visibility within Sun Microsystems' Java Community Source Process, and within the Object Management Group (OMG) in particular, would also constitute an excellent vehicle for these efforts.

Finally, consideration will be given, during the course of the project, to the identification of specific applications areas for the Mikado technology. Several potential areas suggest themselves already:

- Web services and wide area workflow in presence of mobility.
- Active network programming and infrastructures.
- Wide area mobile network control and management.

In all these areas, basic technology developed by the Mikado project should be directly exploitable and should enable a host of new services. All these areas are central for the business of a telecom operator and the presence of France Telecom in the Mikado consortium will facilitate the project investigations in the matter.

9. Workplan

9.1 General description: Overall project structure

The project is organised around three technical work-packages (WP1-WP3) and one organisational work-package (WP4):

- WP1: Programming Models
- WP2: Specification and Analysis
- WP3: Virtual Machine Technology and Language Support
- WP4: Project Co-ordination, Dissemination and Evaluation

These work-packages are intended to run in parallel over the duration of the project. WP4 plays a co-ordination and monitoring role with respect to the activities of the three technical work-packages.

Amongst the three technical work-packages, WP1 plays a central role and will be the subject of close collaborative effort amongst the partners. It will both drive the work of, and receive feedback from, WP2 and WP3. WP2 will work on accompanying specification and static analysis techniques for the modelling work of WP1, and WP3 will cover implementations aspects related to the work of both WP1 and WP2.

The three technical work-packages will run in parallel with careful interaction between the different activities. Due to the inherently speculative nature of the research, results will be refined and extended in an iterative process. The first year of the project will comprise further critical analysis of the state of the art in the areas of global computing, distributed programming models and languages, and virtual machine technologies, the analysis of key programming requirements for global computing, and the development of a preliminary version of the project's core programming model (core programming model release 0) together with a simple type system (type systems release 0). The second year of the project will comprise the development of a first version of the project's programming model (core programming model release 1), together with associated type systems (type systems release 1) and co-inductive proof techniques, and the development of first software prototypes (core virtual machine software framework and simple domain-based programming language). The third and last year of the project will comprise the refinement of the programming model (core programming mode release 2), of type systems and proof techniques (type systems release 2), the development of specification logics associated with the Mikado programming model, and the development of enhanced software prototypes (including domain-specific virtual machine patterns and frameworks, and extensions to object-oriented and functional languages).

Given the long-term research nature of the project, the principal risks to the project are mainly concerned with the lack of scientific breakthroughs ($risk_1$), problems of co-operation ($risk_2$), or the loss of a partner ($risk_3$). $Risk_1$ is lessened by the overall strength and track record of the consortium. $Risk_2$ is similarly lessened by the strong links that already exist between the various partners. Finally, $risk_3$ is minimized by the nature of the consortium and, in particular, the overlapping and complementary skills of the partners (For more details on the consortium, see Appendix A). We are therefore confident we can meet the goals of the project and we can avoid such potential pitfalls.

• Workpackage list

Work-package No[3]	Workpackage title	Lead contractor	Person-months[4]	Start month[5]	End month[6]	Deliv-erable No[7]
WP1	Programming model	INRIA	104 / 116	1	36	D1.1.1 D1.1.2 D1.2.0 D1.2.1 D1.2.2 D1.3.1 D1.3.2
WP2	Specification and Analysis	UOS	110 / 131	1	36	D2.1.0 D2.1.1 D2.1.2 D2.1.3 D2.1.4 D2.2.1 D2.2.2 D2.3.1 D2.3.2
WP3	Virtual machine technology and language support	DSI-UF	131 / 162	1	36	D3.1.0 D3.1.1 D3.1.2 D3.1.3 D3.2.1 D3.2.2
WP4	Project coordination, dissemination and evaluation	INRIA	19 / 29	1	40	D4.1 D4.2 D4.3 D4.4 D4.5 D4.6 D4.7 D4.8 D4.9 D4.10 D4.11

[3] Workpackage number: WP 1 – WP n.
[4] The number of person-months allocated to each workpackage: funded / total.
[5] Relative start date for the work in the specific workpackages, month 1 marking the start of the project, and all other start dates being relative to this start date.
[6] Relative end date, month 1 marking the start of the project, and all ends dates being relative to this start date.
[7] Deliverable number: Number for the deliverable(s)/result(s) mentioned in the workpackage: D1 - Dn.

9.3 Workpackage descriptions

Workpackage 1: Programming Model

WP1 is concerned with the definition of a parametric programming model for global computations, especially dealing with aspects related to mobility and migration. .Results from this workpackage will be used as inputs for the work in WP2 and WP3.

The first phase of this work-package will consist in a detailed comparative study of recent models for distributed and mobile computing, and in their assessment in view of expected programming requirements of global computing systems and applications. The second phase will define a parametric programming model for global computing, based on the notion of domain. .This second phase will also investigate relations between the Mikado migration and mobility model with other well-known approaches to these issues. These phases will be carried out in two different tasks.

The goal of this work-package is an ambitious one. However, the different partners from the project have a strong background in both the theoretical and practical aspects of distributed computing. Also, some of the partners have already engaged in collaborative efforts in trying to analyse requirements for distributed mobile computing in general, and in analysing the state of the art in formal models for distributed mobile computing. These efforts, which will immediately help in bootstrapping the activity of this work-package, show promising directions of research which lead us to believe that the goal is indeed within reach.

It is intended that the migration and mobility model defined by the project in WP1 should subsume the different models studied in the literature, and that, together with the type systems developed in WP2, it can be refined into a cohesive set of complementary models (typically dealing with different aspects of global computing such as security, mobility, fault-tolerance, etc). A key aspect of the programming model and its refinements is that they should allow for the construction of provably correct, or at least formally analysable systems, either applications that conform to the model, or implementations of various aspects of the models in the form of technologies for use in virtual machines that respect the model.

Task 1.1 : State of the Art Analysis

Task 1.1 will perform a review and critical analysis of the current state of the art in concurrent and distributed programming models. Models for study include calculi and languages such as the π-calculus, the spi-calculus, the Join calculus, the Ambient calculus, KLAIM, Nomadic Pict, CHOCS, Facile, Obliq, DiTyCO, etc. Also included in this study will be execution models underlying existing or experimental distributed technology, such as those based on Java and CORBA, including their ad-hoc extensions for mobility, for components (Enterprise Java Beans, CORBA components), for large-scale computing (e.g. Globe), for flexibility and adaptation (e.g. Flexinet, OpenORB, DynamicTAO),etc.

Criteria for analysing and comparing the models will be defined during the course of the work, and will typically be driven by requirements arising from global computing (management of domains, openness, mobility, security, failure detection and management, etc). The analysis will seek to identify elements common to the different models, and to characterise the nature and consequence of their particular formalisation in each model.

The analysis of the state of the art will be completed with an analysis of the key requirements for a global computing programming model. Key considerations will include : different forms of global computing domains, openness, mobility, security, failure detection and management, different forms of interaction and control, levels of abstraction and transparency. This analysis should confirm and substantiate the key insight behind Mikado, namely that a global computing environment can best be understood (modelled and designed) as a set of domains.

Task 1.2 : Domain-based programming model

Task 1.2 will investigate domain-based programming, focusing in particular on issues involving mobility and migration between domains. Considering the diverse requirements existing in a global computing environment, it is unlikely that a unique programming model will emerge that can be used in all global computing situations. Rather, Mikado takes the view that specific situations will emerge as part of specific domains, i.e. system partitions where common sets of behavioral constraints, rules and policies apply. In such a context, work will proceed at three levels:

- the study of a parametric model that captures the notion of domain, and generic constructs dealing with mobility and migration, in the form of distributed process calculus in the tradition of the π-calculus and its variants;
- the study of instances calculi, that can be introduced for different domain types, e.g. by means of specific type systems, that refine the generic core calculus into more specialised programming models, well-adapted to the constraints and semantics of the chosen domain types.
- The study of other distributed calculi, based on already established approaches pursued by the different partners, but informed by the joint work on the parametric model and its instances.

Task 1.2 will interact with task 2.1 to refine the core programming model and to deal with specific domain constraints such as e.g. security and dependability.

Among the studies conducted by task 1.2, we find the study of the relations between Mikado's migration model (together with its domain-specific refinements) and other distributed process calculi..

We expect Mikado's core programming model to be expressive enough to allow a direct and faithful simulation of several recent distributed process calculi. In particular, we hope to characterise as specific kinds of domains the different notions of localities or ambients which appear in these calculi.

We intend to develop domain-based programming in Mikado so that it remains compatible with other programming standards such as object-oriented and functional programming. The combination of different programming language paradigms is a research theme on its own, however we expect to make progress on this task thanks to the insight gained from numerous studies which have related functional and object-oriented programming with the π-calculus, which have studied object-oriented and object-based formal calculi and their relations with the π-calculus and the lambda calculus, and from recent language experiments such as JoCaml, KLAIM, DiTyCO and Nomadic Pict, which combine distributed, object-oriented and functional programming.

Workpackage description: WP1 - Programming Model

Workpackage number :	WP1 – Programming Model					
Start date or starting event:	T0					
Participant :	INRIA	FTRD	DSI-UF	UOS	FFCUL	Total
Person-months per participant (funded):	37	6	30	26	5	104
Person-months per participant (total):	37	6	30	32	11	116

Objectives

- To define a prametric programming model for domain-based global computing.
- To study different domain-based distributed programming models.
- To study the relationships between the Mikado programming models and object-oriented and functional programming models.

Description of work

The work carried out in this work-package will be organised into three main tasks:

- Task 1.1: State of the art analysis: this task will perform a review and critical analysis of the current state of the art in distributed and mobile programming models. This task will also analyse the key requirements for a global computing programming model, including such considerations as openness, mobility, security, failure detection and management, forms of interaction and control, forms of system partition and structuring, levels of abstraction and transparency.
- Task 1.2 Domain-based programming model: this task will develop a parametric model for global computing based on domains and study its relations with other distributed models and programming paradigms

Deliverables

D1.1.1 Analysis of formal models of distribution and mobility: state of the art.

D1.1.2 Requirements for a programming model for global computing.

D1.2.0 Core programming model, release 0.

D1.2.1 A parametric model of migration and mobility, release 1.

D1.3.1 Study of alternate distributed models, release 1.

D1.2.2 A parametric model of migration and mobility, release 2.

D1.3.2 Study of alternate distributed models, release 2.

D1.3.3 Simulating distributed process calculi. A study of the relations between Mikado's domain-based programming model and alternative distributed process calculi.

Milestones and expected result

Year 1 : T0+6: production of deliverable D1.1.1 - T0+9: production of deliverable D1.1.2 – T0+12 : production of deliverable D.1.2.0

Year 2 : T0+21: production of deliverable D1.2.1 - T0+24: production of deliverable D1.3.1

Year 3 : T0+36: production of deliverable D1.2.2 - T0+36: production of deliverables D1.3.2, D1.3.3

Workpackage 2: Specification and Analysis

WP2 is concerned with the definition of Specification and Analysis technologies for the project's programming models. This will range from the development of type systems and static analysis techniques for expressing constraints on concurrency, mobility and resource access for the underlying execution model, to providing proof technologies for assuring that mobile code, and more generally distributed systems, conform to predefined behavioural specifications. The latter will require the definition of novel co-inductive techniques for comparing the distributed behaviour of systems and the elaboration of new specification logics for expressing interesting partial views of systems and programming paradigms.
The development of WP2 will be informed by progress in WP1; in particular the choice of analysis techniques and type systems to be developed will rely heavily on instances of domain calculi which emerge. There will be some time lag between the two workpackages and so much of the work in WP2 will be predicative. We will investigate and develop concepts which in principle will be applicable to the eventual Mikado programing models; only in the later stages of the project will integration be attempted, between the analysis techniques of WP2 and the dominant programming models of WP1. Nevertheless we also hope that experience in WP2 will in turn impact on the languages, and language extensions, developed by WP3.

This Work-package will divide naturally into three interdependent Tasks : Task 2.1 is concerned with the study of static analysis and type systems for the Mikado related programming models; Task 2.2 is concerned with the study of new co-inductive techniques; Task 2.3 is concerned with the definition of new specification logics associated with these programming model.

Task 2.1 Static Analysis and Type Systems

Task 2.1 will first perform a review and critical analysis of the current state of the art in type systems and static analysis of models and languages for distributed mobile computing. Then, on the basis of this analysis, it will develop type systems appropriate to the emerging programming models of Mikado.

The recent work on type systems for object-oriented programming languages and type systems for the π-calculus, the join-calculus, the Dπ-calculus, the spi-calculus etc, as well as the work on static analysis of the π-calculus, of concurrent object-oriented languages and of actor languages will provide a starting point. However new insights will be required in order to cope with the range of domain interpretations, and their operational significance which we envisage within the project (domains embodying e.g.: security barriers and access control points, fault confinement zones and failures modes, resource locations, naming contexts, communication regions, technologically homogeneous sub-systems, software component containers, etc). This task will probably not converge on a single type system or static analysis technique. Instead, it is likely that, due to the nature of the different topics amenable to static analysis in distributed computations, several such systems or techniques will be elaborated. This task will in particular focus on:

- Capturing constraints of concurrency and mobility, for example, to prevent deadlocks and interaction errors.
- Capturing, and implementing, type-theoretic formulations of "proof-carrying code".
- Capturing constraints inherent in different facets of distributed mobile computing, notably with respect to security and dependability (e.g. failure detection).
- Capturing dynamic behavioural constraints, to ensure that only safe and authorised agents will use specific resources. Work on type systems will also cover classical issues of type checking

and type inference, with a view towards practical application in virtual machine and language experiments carried out in WP3.

Task 2.2: Coinductive Proof Techniques

The behaviour of system components depends on how they affect their environment, which may in turn be governed by interaction paradigms currently in force. Well-established theories of concurrency, applicable to simple (undistributed) processes, encode these effects as ``actions" which components may perform, in the presence of a co-operating computing environment. This leads to co-inductive proof techniques for establishing that components have similar behaviour; these techniques are usually based on bisimulations, which treat components as high-level action transducers, and compare their ability to perform similar actions, at similar points in time. The challenge here is to develop similar proof technologies for distributed systems, where components are to be embedded in wide-area systems whose co-operation is to be untrusted, where mobile code is interchanged with unknown principals, and active agents demand attention. Component behaviour will still depend on the ``actions"
they can perform but the nature of these actions are no longer clear, nor when we should deem them to be possible; certainly they will depend on the current type environment, which in turn may evolve over time. Code mobility and active agents automatically entail higher-order activity; currently there are no existing higher-order co-inductive theories applicable to active agents; the success of Mikado will require the development of such theories, moreover theories which are amenable to implementation. This task will in particular focus on :

- Relativised component actions, where the ability to perform an action depends, at least, on the type environment in which the component is active.
- Applicable higher-order versions of bisimulations, which correctly capture the effects of mobile code and agent mobility.
- Distilling example "proof skeletons" applicable to particular programming paradigms, such as server-client or peer-to-peer communication.

Task 2.1, on type systems, will feed into this Task, as we envisage types to be our main mechanism for parameterising component activity. Task 2.2 will in turn feed into WP3, as the chosen language constructs will depend on the availability of viable proof techniques.

Task 2.3: Specification Logics

This is the most speculative part of WP 2. Rather than capture component behaviour in its entirety, it is often preferable to describe, or indeed prescribe, partial views of it's behaviour. How will a component behave in a given environment, when subjected to specific stimuli ? How will a specific piece of mobile code evolve when uploaded to a site having particular properties ? Can we guarantee that it will never access forbidden parts of the host domain? With a given protocol for peer-to-peer interaction can we demonstrate that one peer can not adversely effect the local behaviour of a companion peer ?
Here we need to design specification logics in which these partial views can be formalised, and proof techniques need to be found for establishing that components, agents, and systems conform to these views. At the moment very little is known about expressing desirable properties of components in distributed systems, and even less about techniques for establishing them. Here the challenge is to distill, from current and emerging practise, significant "schemas of behaviour", and to subsequently design a specification language in which these schemas can be expressed and verified. The modal μ-calculus and temporal logics such as CTL have performed this role in standard process theory.

However to specify and describe properties of components in wide-area distributed systems, in the presence of phenomena such as untrusted agents, site failure, unknown communication latency, will require much more sophisticated, and as yet unknown, language features.

Workpackage description: WP2 - Specification and Analysis

Workpackage number :	WP2 – Specification and Analysis					
Start date or starting event:	T0					
Participant number:	INRIA	FTRD	DSI-UF	UOS	FFCUL	Total
Person-months per participant (funded):	38		35	26	11	110
Person-months per participant (total):	38		35	32	26	131

Objectives

Objectives

- To develop type systems a static analysis techniques appropriate to the emerging programming models of Mikado.
- To develop co-inductive techniques for comparing and analysing the distributed behaviour of systems.
- To develop specification logics for the description and verification of behavioural constraints and properties.
-

Description of work

The work carried out in this work-package will be organised into three
main tasks :

- Task 2.1: Static analysis and type systems: this task will perform a review of the current state of the art on type systems and static analysis for programming languages and distributed process calculi, and it will define type systems for the Mikado programming model for capturing constraints of concurrency and mobility, security and dependability.
- Task 2.2: Co-inductive proof techniques: this task will develop co-inductive bisimulation-based proof techniques for the formal comparison of components behaviour in a global computing environment.
- Task 2.3: Specification logics: this task will develop specification logics for the description and verification of behavioural constraints and properties of components in a global computing environment, i.e. in the presence of phenomena such as un-trusted agents, site failures, unknown communication latencies, etc.
-

Deliverables

D2.1.0: Survey of Flat net models: behavioral thoery and type systems
D2.1.1 : Type systems and static analysis for mobile and distributed computing : state of the art.
D2.1.2 : Type systems for Mikado inspired mobility and security programming concepts.
D2.1.3 : Resource policies expressed and checked as type systems
D2.1.4 : Type systems for open networks, using the Mikado models.

D2.2.1 : Co-inductive proof techniques.
D2.2.2 : Proof methodologies in a distributed setting.

D2.3.1 : Modal and temporal logics for distributed behaviour.
D2.3.2 : System behaviour and reasoning in the presence of failures.

Milestones and expected result

Year 1 : T0+6 : production of deliverable D2.1.1
Year 2 : T0+21 : production of deliverables D2.1.2, D2.2.1 - T0+24 :production of deliverables D2.1.0, D2.1.3
Year 3 : T0+30 : production of deliverable D2.2.2 - T0+33: production of deliverable D2.3.1 - T0+36 : productionof deliverables D2.1.4, D2.3.2

Workpackage 3: Virtual machine technology and language support

WP3 is concerned with the embodiment of the Mikado programming models developed in WP1 and WP2 in concrete programming technologies. Work in WP3 will thus be concerned with the development of several prototypes, including:

- Virtual machine technology to support WP1's core programming model together with WP2 typing schemes.
- Languages features and language experiments supporting WP1's core programming model and taking advantage of type systems and static analysis tools developed by WP2.

This work-package thus divides into two inter-dependent tasks:

- Task 3.1 is concerned with the development of virtual machine technology;
- Task 3.2 is concerned with the development of programming language technology.

Task 3.1 Virtual machine technology

Work in this task will be concerned with the development of (provably correct) software technology implementing the core Mikado programming model and its specializations.

Task 3.2 will follow the same reasoning as WP1 Task 1.2 and will develop its virtual machine technology as a two-tiered structure:

- A core software framework supporting recurring patterns for the implementation of different domain-based distributed programming languages. This software framework will consist of a number of core design patterns supporting the primary concepts of domains, binding of names, topology, &.
- Specializations of these core patterns corresponding to specific domain types (security, mobility, fault-tolerance, ..). These domain-specific patterns implement refinements of the core programming model as described in Task 1.2.

The approach in Task 3.1 is predicated on the possibility to identify generic design patterns for implementing Mikado's domain-based programming models. This approach to building execution support for the Mikado programming model should offer several benefits:

- An identification of common virtual machine components occurring in the execution structure of different distributed programming languages (see Task 3.2 below).
- An assessment of existing distributed software platforms such as Java, CORBA and COM and a comparison with our choice.
- A study of the possible reuse of correctness properties that would guarantee that specialisation preserves existing correctness properties at both the model and the implementation levels. Evidently, specialised models and patterns will add new properties, which will require new proofs. It should be noted that the work described here is ambitious. Provably correct component-based implementations remains a subject of active research but the work of Tasks 2.2 and 2.3 should provide important input and guidelines to this effort.

Work in Task 3.1 will comprise an analysis of the state of the art in virtual machine technology for distributed and mobile computing and the development of the core software framework and domain specific patterns. In terms of programming environments with underlying formal models of distributed and mobile computing, a number of implementations already exist: JoCaml, Ambit, Nomadic Pict, X-Klaim, DiTyCO, Kali Scheme, etc. One of the aims of this task is to take stock of these existing virtual machines (along with their associated languages, virtual machines and execution structures) and analyse them in terms of organisation, functions, and possible support for (in Mikado terms) domain-specific constraints. Other distributed system technologies exist, that support limited domain-like structures, but without a formal underlying model; for example, EJB containers and servers, CORBA component containers, class-loaders in the Java virtual machine, etc. A selection of these technologies will also be analysed, from the point of view of their organisation and functions, and of the kinds of domains that they support. The core framework and the domain-specific patterns can be rendered in different programming languages. Key considerations that will be examined during this task include:

- Proofs of correctness of the core framework and domain-specific patterns, and their assemblage with respect to the programming models of WP1.
- Truly distributed implementations, with different forms of mobility, depending e.g. on whether names or agents are moved.
- Type-driven optimisations (for instance, linear or receptive channels are amenable to more efficient implementations than standard communication channels).
- Intermediate code representations (e.g. byte codes, slim binaries, typed assembly languages) and associated support for proof-carrying code.
- Interoperability with non- Mikado systems.

Task 3.2 Language experiments

Studies in WP1 and WP2 will together define formal programming models for global distributed computing. However a programming model is too minimal to constitute a usable programming language (contrast, for instance, the ML programming language and its underlying programming model, a typed variant of the lambda-calculus). Task 3.1 will study programming idioms based on the programming model resulting from WP1 and WP2. In particular, this work-package will study:

- Using Mikado calculi studied in WP1 as the basis of simple programming languages, with the goal to study in native form new programming constructs resulting from the Mikado programming models.
- The extension of other programming languages that have been developed by partners to take into account object-oriention and functional programming languages and to take into account domain-based constructs and type systems which constitute the Mikado programming model. The theoretical basis for these extensions will be the study of relations between the Mikado programming model and other programming paradigms undertaken in Task 1.3.

With the help of results on more theoretical aspects tackled in Task 1.3, this task should consider various issues which are typical of programming language design experiments. These include for example: inclusion of primitive and abstract data types, inclusion of imperative features (e.g. assignments), exception handling, I/O constructs, language modularity constructs (in particular interaction between different forms of encapsulation and modularity in language extensions), interaction, concurrency and superimposition constructs, etc. Prototyping work in this task will make use of the results of Task 3.2 on virtual machine technology. In particular, this task will consider, where appropriate (i.e. in experiments dealing with extensions to virtual-machine-based programming language implementations), the coupling between the virtual machine technology developed in Task 3.1 and language-specific virtual machines and execution structures. This coupling could take different forms, depending on the results in Task 3.1: simple interfacing of a language-specific library in the most favourable situation, complete overhaul of an existing language environment in the least favourable one.

Workpackage description: WP3 - Virtual Machine Technology and Language Support

Workpackage number :	WP3 – Virtual Machine Technology and Language Support					
Start date or starting event:	T0					
Participant number:	INRIA	FTRD	DSI-UF	UOS	FFCUL	Total
Person-months per participant (funded):	38	15	38	20	20	131
Person-months per participant (total):	38	15	38	23	48	162

Objectives

- To develop virtual machine technology for the efficient execution of the Mikado programming model.
- To develop programming language idioms conforming to the Mikado programming model.

Description of work

The work carried out in this work-package will be organised into two main tasks :

- Task 3.1: Virtual machine technology: this task will perform a review and critical analysis of virtual machine and execution structures for distributed, mobile languages, and will develop a distributed virtual machine technology for the Mikado programming model according to a 2-tier structure (core software framework, domain-specific patterns).
- Task 3.2: Language experiments: this task will implement programming language idioms to support Mikado domain-based programming models. It will comprise: the definition and implementation of simple domain-based programming languages based on Mikado calculi and of extensions to existing object-oriented and functional languages such as Java, ML.

Deliverables

D3.1.0 : Virtual machine technology: Core software framework, v0

D3.1.1 : Analysis of distributed execution structures: state of the art.

D3.1.2 : Virtual machine technology: Core software framework, v1.

D3.1.3: Virtual machine technology: Core software framework v2.

D3.2.1: Language experiments v1: Simple calculi as programming languages

D3.2.2: Language experiments v2: Extensions to object-oriented and functional languages

Milestones and expected result

Year 1 : T0+6 : production of deliverable D3.1.1 – T0+12 : production of deliverable D.3.1.0

Year 2 : T0+24 : production of deliverable D3.1.2,

Year 3 : T0+30: production of deliverable D3.2.1, T0+36 : production of deliverables D3.1.3, D3.2.2

Work-package 4: Project coordination, dissemination and evaluation

This work-package is responsible for the overall co-ordination of the project, for its various dissemination activities and for continuous evaluation and assessment of the project results.

Project co-ordination will be primarily effected at the work-package level. Each work-package will have a work-package leader, responsible for defining and co-ordinating activities within the work-package. The project coordination will be aided by the Project Co-ordination Committee (PCC). The main responsibilities of the PCC include monitoring the progress of the project with respect to the overall work-plan, to ensure timely delivery of results, to identify potential problems, to take decisions regarding the project's technical and co-ordination activities, and to identify opportunities for the project's work to be promoted and disseminated. These roles are defined in detail in section 9.7 below.Dissemination will play an important role in the project. We will target publication in high quality international conferences and journals, including newly emerging initiatives around the theme of global computing. We plan to host project workshops annually that will be open to IST-funded projects, and especially those from the FET Global Computing programme. These workshops will also be open to researchers in Europe and world-wide, although numbers will be limited to ensure discussion and cross-fertilisation of results. In addition, we plan to organize international workshops affiliated to major conferences.

We will finally encourage the dissemination of the more mature software prototypes developed by the project under open-source licences. It is expected that most software prototypes developed by the project will be made available under open-source licences on the project Web site (see section 9.7).

Assessment and evaluation of the project results will be effected continuously during the lifetime of the project and will be one of the prime responsibility of the PCC. In addition to continuous monitoring of the project results, the PCC will be responsible for producing yearly project assessment and evaluation reports.

These different activities translate into two main tasks for this workpackage: Task 4.1 is concerned with the overall management of the project and dissemination activities; Task 4.2 is concerned with the assessment and evaluation of the project.

Task 4.1 : Management and dissemination

Task 4.1 comprises the management, coordination and dissemination activities of the PCC. Management and coordination within the project will be the responsibility of Workpackage Leaders, of the Project Director and of the Project Coordination Committee.

Workpackage leaders are responsible for coordinating the technical work carried out in technical workpackages

The project Director is responsible for the overall management of the project, in line with decision taken by the PCC, and for overseeing and coordinating all technical work according to the project work-plan and timetable.The project Director also oversees the evaluation and assessment of the project results (see Task 4.2 below).

The *Project Coordination Committee (PCC)* is the main decision-making body of the project, and is responsible for the technical and strategic directions of the project, as well the financial and logistical management of the project (see section 9.7 below).

As part of its activities, the PCC will also manage and organize dissemination activities of the project. These will comprise:

- Publications at major international conferences and in high-quality international scientific journals. Target journals typically include: Theoretical Computer Science, Mathematical Structures in Computer Science, Higher-Order and Symbolic Computation, ACM Trans. on Programming Languages and Systems, Information and Computation, IEEE Trans. on Software Engineering, Distributed Computing, Software Practice and Experience. Target international conferences typically include: ACM POPL, ACM PLDI, ICALP, IEEE LICS, CONCUR, ETAPS, ACM OOPSLA, IEEE ICDCS, IFIP/ACM Middleware.

- Maintenance of a project Web site including a public area where the different report s, papers and software produced by the project can be found.
- Organization of yearly project open workshops.
- Organization of open workshops, in affiliation with major international conferences (e.g. ETAPS, ICALP or CONCUR).
- Organization of an international summer school on mobility and programming for global computing for PhD students and people from industry.
- Dissemination of the software produced by the project, where applicable, under open-source licence to facilitate and encourage experimentation with Mikado ides and results.

Task 4.2 : Assessment and evaluationAssessment and evaluation within the project will be conducted primarily by the PCC, which is responsible for the overall quality management of project productions (review of technical reports and papers, value assessment of ideas and developments, approval of project deliverables, etc) and for the production of yearly assessment and evaluation reports. The latter reports will provide:

- A brief review of the project results and of its dissemination activities.
- An assessment of the quality and significance of the projects results.
- An assessment of the progress of the project, overall and per technical workpackage.

In addition, a project Advisory Board will be established, composed of invited personalities external to the project. The main role of this board is to review the scientific quality and relevance of the project productions (see section 9.7).Concerning the assessment of the project progress towards its scientific and technical objectives, deliverable D1.1.2 on “Requirements for a programming model for global computing”, will play a crucial role for it will provide the project with a precise set of yardsticks against which to measure the achievements of the technical workpackages. Yearly assessment and evaluation reports, as well as the final report of the project, will in particular measure the coverage of the requirements in D1.1.2 achieved in the course of the project. In turn, deliverable D1.1.2 will be updated in the course of the project to reflect the potentially changing perception of the project on global computing requirements and their relative priorities. Key high-level criteria for assessing the scientific and technical success of the project are as follows:

- Ability to capture in the Mikado programming model key concepts for global distributed computing, especially accounting for the different forms of domains (security, failure, administration, etc) that occur in a highly dynamic global computing environment. As a side effect, the Mikado programming should be shown to subsume (eg ability to faithfully simulate) current distributed process calculi proposals.
- Ability to capture in the Mikado type systems key constraints and properties that need to be enforced or statically verified in global computations (e.g. absence of security and communication failures).
- Ability to characterize program equivalence and simulation relations in presence of higher-order features which are characteristic of global computing programs.
- Ability to leverage the Mikado programming model into full fledge programming languages for global computing, including its effective combination with established programming paradigms such as object-oriented and functional programming.
- Ability to leverage the more theoretical results of the project into effective software technologies (virtual machine substrate and actual programming languages).

Workpackage description: WP4 - Project Co-ordination, Dissemination and Evaluation

Workpackage number :	WP4 – Project Co-ordination and Dissemination					
Start date or starting event:	T0					
Participant number:	INRIA	FTRD	DSI-UF	UOS	FFCUL	Total
Person-months per participant (funded):	12	2	5	0	0	19
Person-months per participant (total):	12	2	5	5	5	29

Objectives

- To manage the project as a whole and to coordinate technical activities within the project
- To plan and organize the dissemination of the project results
- To assess and evaluate the progress of the project and the scientific quality and relevance of its results

Description of work

The work carried out in this workpackage will be organized into 2 main tasks :

- Task 4.1: Management and Dissemination: this task will comprise the management and coordination activities for the project as a whole, and will plan and implement the main dissemination activities of the project.
- Task4.2: Assessment and Evaluation: this task will conduct periodic assessments and evaluations of the project progress and results.

The work carried out in this task will be the prime responsibility of the project Director, of the Workpackage Leaders, and of the Project Coordination Committee, with the help of the project's Advisory Board for the evaluation and assessment of the project scientific and technical results.

Deliverables

D4.1 Project Presentation

D4.2 Dissemination and Use Plan

D4.3 Self-assessment Plan

D4.4 Annual Report Year 1 (containing managerial and factual information concerning the project)

D4.5 Dissemination and Evaluation Report Year 1 (containing an assessment of the project progress and results as well as information concerning dissemination activities undertaken by the project)

D4.6 Annual Report Year 2

D4.7 Dissemination and Evaluation Report Year 2

D4.8 Annual Report Year 3

D4.9 Dissemination and Evaluation Report Year 3

D4.10 Final Project Report (describing the main project results and achievements, together with key factual, dissemination and exploitation information concerning the project)

D4.11 Technology Implementation Plan (indicating how partners intend to exploit project results)

Milestones and expected result

Year 1 : T0+3 : deliverable D4.1 - T0+6 : deliverable D4.2 – T0+9 : deliverable D4.3 - T0+12 : deliverables D4.4 and D4.5 -- Year 2 : T0+24 : deliverables D4.6 and D4.7 -- Year 3 : T0+36 : deliverables D4.8 and D4.9 -- T0+12, T0+24, T0+36: annual progress reports. Year 4: T0+40: deliverables D4.10 and D4.11

9.4 Deliverables list

Deliverables list

Del. no.	Deliverable name	WP no.	Lead	Estimated person-months	Del. type*	Security**	Delivery (proj. month)
D4.1	Project Presentation	4	INRIA	0.5	Report	Public	3
D1.1.1	Analysis of formal models of distribution and mobility: state of the art	1	INRIA	6	Report	Public	6
D2.1.1	Type systems and static analysis for mobile and distributed computing: state of the art	2	UOS	6	Report	Public	6
D3.1.1	Analysis of distributed execution structures: state of the art	3	DSI-UF	6	Report	Public	6
D4.2	Dissemination and Use Plan	4	INRIA	1	Report	Restricted	6
D1.1.2	Requirements for a programming model for global computing	1	INRIA	12	Report	Public	9
D4.3	Self-assessment Plan	4	INRIA	1	Report	Restricted	9
D1.2.0	Core programming model, v0	1	INRIA	12	Report	Public	12
D2.1.0	Survey of flat net models	2	UOS	9	Report	Public	24
D3.1.0	Virtual machine technology : Core software framework, v0	3	DSI-UF	18	Prototype	Restricted	12
D4.4	Annual project report Year 1	4	INRIA	1	Report	Restricted	12
D4.5	Dissemination and Evaluation report: Year 1	4	INRIA	1	Report	Restricted	12
D1.2.1	Parametric migration model, v1	1	INRIA	18	Report	Public	21
D2.1.2	Type systems for mobility and security	2	UOS	24	Report	Public	21
D2.2.1	Co-inductive proof techniques	2	UOS	21	Report	Public	21
D1.3.1	Study of alternate models	1	INRIA	12	Report	Public	24
D2.1.3	Resource policies as type systems	2	UOS	9	Report	Public	24
D2.3.1	Modal and temporal logics for distributed behaviour	2	UOS	9	Report	Public	33
D3.1.2	Virtual machine technology: Core software framework, v1	3	DSI-UF	30	Prototype	Restricted	24
D3.2.1	Language experiments v1: Simple calculi as programming languages	3	DSI-UF	25	Prototype	Restricted	30
D4.6	Annual project report Year 2	4	INRIA	1	Report	Restricted	24
D4.7	Dissemination and Evaluation report: Year 2	4	INRIA	1	Report	Restricted	24
D2.2.2	Proof methodologies	2	UOS	15	Report	Public	30
D1.2.2	Parametric migration model, v2	1	INRIA	20	Report	Public	36

D1.3.2	Study of alternate models, v2	1	INRIA	30	Report	Public	36
D2.1.4	Type systems for open networks	2	UOS	10	Report	Public	36
D2.3.2	System behaviour and reasoning in the presence of failures	2	UOS	18	Report	Public	36
D3.1.3	Virtual machine technology: domain-specific patterns	3	DSI-UF	35	Prototype	Restricted	36
D3.2.2	Language experiments v2: Extensions to object-oriented and functional languages	3	DSI-UF	50	Prototype	Restricted	36
D4.8	Annual project report: Year 3	4	INRIA	1	Report	Restricted	36
D4.9	Dissemination and Evaluation report: Year 3	4	INRIA	1	Report	Restricted	36
D4.10	Final project report	4	INRIA	1	Report	Public	40
D4.11	Technology implementation plan	4	INRIA	1	Report	Restricted	40

* *A short, self-evident description e.g. report, demonstration, conference, specification, prototype…*

***Int.*	*Internal circulation within project (and Commission Project Officer if requested)*
Rest.	*Restricted circulation list (project partners) and Commission PO only*
IST	*Circulation within IST Programme participants*
FP5	*Circulation within Framework Programme participants*
Pub.	*Public document*

9.5 Project planning and time table (GANTT Chart)

See Appendix D.