MIKADO Global Computing Project
IST-2001-32222

Mobile Calculi Based on Domains

The Mikado Global Computing Project: Presentation

An executive summary of the Mikado project

Title :	The Mikado Global Computing Project: Presentation
Editor :	J.B. Stefani (INRIA)
Authors :	Mikado Consortium
Classification :	Deliverable D4.1, Public
Reference :	MR/WP4/1
Version :	1.0
Date :	April 2002

Abstract

This document presents a brief summary of the Mikado project: its scientific and technical objectives, its innovative aspects and expected results, its organization, and its partners.

Contents

1 Overview

The Mikado project is part of the IST FET pro-active initiative on Global Computing. It has been launched in January 2002 for a duration of 36 months. The Mikado consortium comprises the following partners: INRIA (France), France Telecom R&D (France), the University of Florence (Italy), the University of Sussex (United Kingdom), and the University of Lisbon (Portugal), with additional contributions from the University of Torino (Italy) and ENST (France). INRIA, the French National Institute for Research in Computer Science and Control, is the coordinating partner. The overall estimated effort for Mikado is 36 man.year.

2 Objectives and Motivations

The overall goal of the Mikado project is to construct a new formal programming model, based upon the notion of domain as a computing concept, which supports reliable, highly distributed and mobile computation, and provides 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" . 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 lay the foundation necessary 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

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

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, 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.

3 Innovation and Expected Results

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, Microsoft .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 [4], Orca [17], Kali Scheme [11], Facile [16], Obliq [7], 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 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 [18, 21], support for large-scale distribution and replication [31], support for real-time and multimedia constraints [15], 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 [6, 5, 19, 22] 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 [25], or some forms of object-based or actor-based calculi (see e.g. [1, 29]). More recently, several formal models for distributed and mobile computing have been proposed, including, for example, the Join calculus [12], the π_{1l}-calculus [2], the Ambient calculus

[9], the Safe Ambient calculus [23], the Seal calculus [32], the Dπ-calculus [20], the D$\lambda\pi$ calculus [33], DiTyCO [24], Nomadic Pict [27], KLAIM [26], Oz [28] (see [10] for a survey of process calculi with localities). Also, various extensions of the Actor model, such as [14], 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 and expected results of Mikado will comprise:

- A core formal programming model 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 [8]. 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.

- 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 domain-based constructs with object-oriented and functional programming languages, and provably correct virtual machine prototypes for the Mikado 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. [13, 30]).

4 Organization and Workplan

The project is organised around three themes: core programming model, specification and analysis, and virtual machine technologies and language support.

The core programming model for global computing will be based on the notion of domain. 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, a programming model for global computing should be envisaged as a two-tiered structure :

- the core programming model captures the notion of domain, and the generic constructs which apply across various domains, in the form of a distributed process calculus;

- specific sub-calculi 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 specification and analysis techniques for the programming model will range from type systems and static analysis techniques for expressing constraints on concurrency, mobility, and resource access for the underlying execution model, to proof techniques 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 to compare the distributed behaviour of systems and the elaboration of new specification logics for expressing interesting partial views of systems and programming paradigms.

The virtual machine technologies and language support will embody the Mikado programming model in concrete programming technologies. We will develop several prototypes, including: virtual machine technology supporting the programming model together with typing schemes; language features and language extensions supporting the programming model and the type systems.

5 Further Information

For further information concerning the Mikado project, consult the project Web site at: `http://mikado.di.fc.ul.pt`.

References

[1] M. Abadi and L. Cardelli. *A theory of objects*. Springer, 1996.

[2] R. Amadio. An asynchronous model of locality, failure, and process mobility. Technical report, INRIA Research Report RR-3109, INRIA Sophia-Antipolis, France, 1997.

[3] J. Bacon, K. Moody, J. Bates, R. Hayton, C. Ma, A. McNeil, O. Seidel, and M. Spiteri. Generic support for distributed applications. *IEEE Computer, March*, 2000.

[4] A.D. Birrell, G. Nelson, S. Owicki, and E. Wobber. Network objects. In *in Proceedings 14th Symp. on Operating Systems Principles (SOSP)*, 1993.

[5] G. Blair, G. Coulson, A. Andersen, L. Blair, M. Clarke, F. Costa, H. Duran-Limon, T. Fitzpatrick, L. Johnston, R. Moreira, N. Parlavantzas, and K. Saikoski. The Design and Implementation of OpenORB v2. *IEEE Distributed Systems Online, vol. 2 no 6, Special Issue on Reflective Middleware*, 2001.

[6] G. Blair, G. Coulson, P. Robin, and M. Papathomas. An architecture for next generation middleware. In *Proceedings IFIP International Conference Middleware '98, Lake District, UK*, 1998.

[7] L. Cardelli. A language with distributed scope. *Computing Systems Vol. 8 No. 1*, 1995.

[8] L. Cardelli. Wide area computation. In *Proc. Automata, Languagese and Programming, 26th International Colloquium, (ICALP'99), J. Wiedermann, P. van Emde Boas, M. Nielsen (eds), Lecture Notes in Computer Science, Vol. 1644*. Springer, 1999.

[9] L. Cardelli and A. Gordon. Mobile ambients. In *Foundations of Software Science and Computational Structures, M. Nivat (Ed.), Lecture Notes in Computer Science, Vol. 1378*. Springer Verlag, 1998.

[10] I. Castellani. Process algebras with localities. In *Handbook of Process Algebra, J. Bergstra, A. Ponse and S. Smolka (eds)*. Elsevier, 2001.

[11] H. Cejtin, S Jagannathan, and R. Kelsey. Higher-order distributed objects. *ACM TOPLAS Vol. 17 No. 5*, 1995.

[12] C. Fournet, G. Gonthier, J.J. Levy, L. Maranget, and D. Remy. A calculus of mobile agents. In *In Proceedings 7th International Conference on Concurrency Theory (CONCUR '96), Lecture Notes in Computer Science 1119*. Springer Verlag, 1996.

[13] C. Fournet, J.J. Levy, and A. Schmitt. An asynchronous distributed implementation of mobile ambients. In *Proceedings of the International IFIP Conference TCS 2000, Sendai, Japan, Lecture Notes in Computer Science 1872*. Springer, 2000.

[14] S. Frolund. *Coordinating Distributed Objects – An Actor-Based Approach to Synchronization.* MIT Press, 1996.

[15] J.B. Stefani G. Blair. *Open Distributed Processing and Multimedia.* Addison-Wesley, 1997.

[16] A. Giacalone, P. Mishra, and S. Prasad. Facile: a symmetric integration of concurrent and functional programming. *Intern. J. of Parallel Programming Vol. 18 No. 2*, 1989.

[17] S. Ben Hassen, H. Bal, and C. Jacobs. A Task and Data Parallel Programming Language based on Shared Objects. *ACM. Trans. On Programming Languages and Systems*, 1998.

[18] R. Hayton, M. Bursell, D. Donaldson, and A. Herbert. Mobile java objects. In *Proceedings IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing (Middleware '98), Davies, Raymond, Seitz (eds), Springer*, 1998.

[19] R. Hayton, A. Herbert, and D. Donaldson. FlexiNet: A Flexible Component-oriented Middleware System. In *Proc. 8th ACM SIGOPS European Workshop on Support for Composing Distributed Applications, Sintra, Portugal*, 1998.

[20] M. Hennessy and J. Riely. Resource access control in systems of mobile agents. Technical report, Technical Report 2/98 – School of Cognitive and Computer Sciences, University of Sussex, UK, 1998.

[21] A. Joseph, J. Tauber, and F. Kaashoek. Mobile computing with the rover toolkit. *IEEE Transactions on Computers: Special issue on Mobile Computing, Vol. 46, No. 3*, 1997.

[22] F. Kon, M. Roman, P. Liu Mao, T. Yamane, L.C. Magalhaes, and R. Campbell. Monitoring, Security and Dynamic Configuration with the dynamicTAO Reflective ORB. 2000.

[23] F. Levi and D. Sangiorgi. Controlling interference in ambients. In *Proceedings 27th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL 2000)*, 2000.

[24] L. Lopes, F. Silva, A. Figueira, and V. Vasconcelos. DiTyCO: An Experiment in Code Mobility from the Realm of Process Calculi. In *Proceedings 5th Mobile Object Systems Workshop (MOS'99)*, 1999.

[25] R. Milner. *Communicating and mobile systems : the π-calculus.* Cambridge University Press, 1999.

[26] R. De Nicola, G.L. Ferrari, and R. Pugliese. Klaim: a Kernel Language for Agents Interaction and Mobility. *IEEE Trans. on Software Engineering, Vol. 24, no 5*, 1998.

[27] P. Sewell, P. Wojciechowski, and B. Pierce. Location-independent communication for mobile agents : a two-level architecture. Technical report, TR-462, Computer Lab, University of Cambridge, Cambridge, UK, 1998.

[28] G. Smolka, M. Henz, and 1995 J. Wurz . *Object-oriented concurrent constraint programming in Oz.* In Hentenrink, Saraswat (eds) Principles and Practice of Constraint Programming, MIT Press, 1995.

[29] C. Talcott. Composable semantics models for actor theories. *Higher-order and symbolic computation, Vol.11, No 3*, 1998.

[30] A. Unyapoth and P. Sewell . 2001. Nomadic Pict: Correct Communication Infrastructures for Mobile Computation. In *Proceedings ACM Int. Conf. on Principles of Programming Languages (POPL)*, 2001.

[31] M. van Steen, P. Homburg, and A.S. Tanenbaum. Globe: A wide-area distributed system. *IEEE Concurrency, January-March*, 1999.

[32] J. Vitek and G. Castagna. Towards a calculus of secure mobile computations. In *Proceedings Workshop on Internet Programming Languages, Chicago, Illinois, USA, Lecture Notes in Computer Science 1686, Springer*, 1998.

[33] N. Yoshida and M. Hennessy. Assigning types to processes. In *15th Annual IEEE Symposium on Logic in Computer Science (LICS)*, 2000.