

# INRIA, Evaluation of Theme ComC

Project-team MIMOSA

04/24/2007

**Project-team title : Migration and Mobility: Semantics and Applications**

**Scientific leader : Gérard Boudol**

**Research center : Sophia-Antipolis**

**Common project-team with : Centre de Mathématiques Appliquées de l'École Nationale Supérieure des Mines de Paris**

## 1 Personnel

### Personnel (03/26/2003)

	Misc.	INRIA	CNRS	University	Total
DR (1) / Professors	1	1		1	<b>3</b>
CR (2) / Assistant Professors		2	1		<b>3</b>
Permanent Engineers (3)					
Temporary Engineers (4)					
PhD Students	3			5	<b>8</b>
Post-Doc.		1			<b>1</b>
<b>Total</b>	<b>4</b>	<b>4</b>	<b>1</b>	<b>6</b>	<b>15</b>
External Collaborators		1			<b>1</b>
Visitors (> 1 month)					

- (1) "Senior Research Scientist (Directeur de Recherche)"
- (2) "Junior Research Scientist (Chargé de Recherche)"
- (3) "Civil servant (CNRS, INRIA, ...)"
- (4) "Associated with a contract (Ingénieur Expert or Ingénieur Associé)"

### Personnel (04/24/2007)

	Misc.	INRIA	CNRS	University	Total
DR / Professors	1	2			<b>3</b>
CR / Assistant Professor		2			<b>2</b>
Permanent Engineer					
Temporary Engineer					
PhD Students	1			3	<b>4</b>
Post-Doc					
<b>Total</b>	<b>2</b>	<b>4</b>		<b>3</b>	<b>9</b>
External Collaborators				1	<b>1</b>
Visitors (> 1 month)				1	<b>1</b>

## Changes in staff

DR / Professors CR / Assistant Professors	Misc.	INRIA	CNRS	University	total
Arrival		2 (MS+OT)			2
Leaving		1 (MS)	1 (SDZ)	1 (RA)	3

**Comments:** Roberto Amadio (RA) moved in September 2005 from a Professor position in Marseilles to one in Paris. This caused the Mimosa project to stop being a joint project with the Laboratoire d’Informatique Fondamentale of Marseilles, and consequently Silvano Dal Zilio (SDZ, CR CNRS) left the Mimosa project (while Roberto Amadio is now an “external collaborator”). Manuel Serrano (MS) moved from a CR position to a DR position at INRIA in 2005. Finally in 2007 Olivier Tardieu (OT) joined us as a CR INRIA, on secondment from the Corps des Mines.

## Current composition of the project-team (04/24/2007):

- Roberto Amadio, Professor, Paris 7 University, External Collaborator.
- Gérard Boudol, DR1, INRIA.
- Frédéric Boussinot, Maître de Recherche, ENSMP.
- Ilaria Castellani, CR1 INRIA.
- Frédéric Dabrowski, PhD student.
- Stéphane Epardaud, PhD student.
- Erick Gallesio, Assistant Professor, Nice-Sophia-Antipolis University, Visitor.
- Marija Kolundzija, PhD student, Torino University.
- Florian Loitsch, PhD student.
- Manuel Serrano, DR2 INRIA.
- Olivier Tardieu, CR2, on secondment from the Corps des Mines.

## Current position of former project-team members (including PhD students during the 2003-2007 period):

- Raul Acosta (PhD student), Associate Professor, Mexico.
- Yannis Bres (temporary engineer), Stensoft.
- Christian Brunette (PhD student), temporary engineer, ESPRESSO team, IRISA.
- Damien Ciabrini (PhD student), Amadeus.
- Silvano Dal Zilio (permanent researcher), CR CNRS.
- Xudong Guan (post-doc), post-doc, Lisbon University, ?
- Ana Matos (PhD student), assistant professor, Instituto Superior Técnico, Lisbon.
- Charles Meyssonier (PhD student), ?
- Vincent Vanackère (PhD student), ?
- Pascal Zimmer (PhD student), post-doc BRICS, Jane Street Capital NY.

## Last INRIA enlistments

- Olivier Tardieu, 2007, CR2 on secondment from the Corps des Mines.

## 2 Work progress

### 2.1 Keywords

concurrent programming, functional programming, mobile code, reactive programming, resource control, secure information flow, web programming.

### 2.2 Context and overall goal of the project

The “flag” of the MIMOSA project is *mobile code*, but we actually have a larger domain of research. First, we conceive mobile code as part of a *concurrent* system: when a mobile agent moves for execution on some machine, it may meet other agents on this machine, and the resulting assembly of agents is performing in a concurrent manner. Then one of our main areas of research was, and still is, concurrent (and mobile) programming models. We are developing in particular the *reactive* style, which evolves from synchronous programming. Next, mobile code must obviously be written in some programming language. Several options are possible here: the idea of “code on demand” was popularized in the context of the JAVA language, but it is not inherently tied to any programming style. In the MIMOSA project we mainly develop the *functional* (and imperative) approach to programming. One of the advantages of this approach is that a clear, formal *semantics* can be given to programs, and this in turn may be used as a basis for various formal analyses of programs. Finally, dealing with mobile code motivates us to tackle *security* (and safety) issues. In this respect, our approach is to develop methods that could be useful from a programmer’s point of view, who would have to develop security-minded software, rather than to design a posteriori verification and certification techniques for executable code – although the difference between the two approaches is not so definite. Then we are looking for effective static analysis methods for safety and security issues, like not consuming too much computing power or inadvertently disclosing confidential information.

Summarizing, the overall goal of the MIMOSA project is to design and implement concurrent (reactive) and functional programming constructs, with the objective of supporting mobile code, and to develop analysis techniques for this programming style, especially regarding safety and security concerns.

Capitalizing on our expertise in models of concurrent and mobile systems, and more generally in semantics, and in concurrent and functional programming, we have progressed during the past four years period in the above-mentioned directions (see below). We are starting to investigate applications. On this way, we recently made a big step by seriously considering the WorldWideWeb context in which mobile code is naturally expected to be useful. This, for us, means designing and implementing a programming language that would allow the programmer to master at a high-level of abstraction the overwhelming amount of technologies involved in web programming.

### 2.3 Objectives for the evaluation period

In our report covering the previous evaluation period<sup>(1)</sup>, we listed (on pages 16-18) 14 research objectives for the 2003-2007 period. Since the format of the report has changed, these are gathered below under three main headings: *safe parallel and reactive programming*, *mobile code and web programming languages*, and *security*. Most of the objectives that we set ourselves have been achieved, as described below, and only two of them have not been tackled. Some of our achievements were not expected or announced in 2003, the most important ones being: in the area of reactive programming, work towards the safe integration of this style of programming into the more classical one of preemptive multithreading (in order to benefit from modern architectures in particular), and work towards a theoretical model of reactive programming; in the area of web programming languages, the design and development of the HOP language, for programming interactive web applications; in the security area, a programming construct for declassifying confidential information, together with the definition of a new security property (akin to non-interference), and the design of a new security type system for imperative and functional programs with declassification.

In the following [Mimosa03] refers to our report for the previous evaluation<sup>(1)</sup>.

---

<sup>(1)</sup> available as <http://www-sop.inria.fr/mimosa/Gerard.Boudol/mimosa-eval03.pdf>

## 2.4 Safe parallel and reactive programming

### 2.4.1 Personnel

Raul Acosta, Roberto Amadio, Gérard Boudol, Frédéric Boussinot, Christian Brunette, Ilaria Castellani, Frédéric Dabrowski, Stéphane Epardaud, Manuel Serrano.

### 2.4.2 Project-team positioning

There is nowadays a renewal of interest in *concurrent programming*, for at least two reasons: one is that multi-core or multi-processor architectures are on the market, and that one would like to take advantage of this at the application programming level. A second reason is that some modern applications, like web servers, are inherently massively concurrent, and here again there is a need for concurrent programming techniques. We notice that in the latter case, many research works advocate the *cooperative* style of programming (including the event-driven model) as a more effective user-level concurrent model than classical multi-threading. The *reactive* model that we are developing follows this style. Reactive programming evolves from synchronous programming *à la* ESTEREL. As such, it relies on concurrent components sharing instants, and synchronizing by waiting and emitting broadcast signals. The reactive model intends to escape from the niche of synchronous programming – namely real-time applications and hardware design –, to provide an alternative to multi-threading. In particular, it intends to be more modular than synchronous programming, allowing concurrent components to be added to a system without introducing causality cycles. Technically, the key for this is to defer the reaction to the absence of a signal to the next instant. The reactive programming style is mainly developed in France, in our group, and in the work of Marc Pouzet and Louis Mandel (at Paris 6 University) on REACTIVEML for instance, with a contribution by Riccardo Pucella at Bell Labs in the context of STANDARD ML. A similar style has been investigated in the context of functional programming (using HASKELL) in Paul Hudak’s group at Yale University, and then by Conal Elliot at Microsoft Research, and also in a JAVA context, with the TRIVENI API by Radha Jagadeesan & al., and FRAPPÉ by Antony Courtney. The synchronous approach is developed by the SYNCHRONE group at Verimag Grenoble (Florence Maraninchi and Nicolas Halbwachs), and, at INRIA, in the AOSTE and EXPRESSO teams.

### 2.4.3 Scientific achievements

Due to the fact that we could not offer a PhD position to Alexander Samarin, the objective (13) described in [Mimosa03] has not been tackled. The goal was to apply reactive programming to the (visual) simulation of physical systems. A demo of our preliminary results on this topic is still available at [http://www-sop.inria.fr/mimosa/rp\\_2004/SimulationInPhysics/index.html](http://www-sop.inria.fr/mimosa/rp_2004/SimulationInPhysics/index.html).

Our objective (12) was about optimized implementations of reactive engines, including the design of byte-code implementations to support migration. This was achieved in the PhD work of Raul Acosta (overlapping with the previous evaluation period) [5, 71], and in the work of Stéphane Epardaud on the implementation of ULM [45, 78, 79]. The latter will be reported upon in more details in the next section. Our objective (14) on ICOBJS was partly covered by the PhD Thesis work of Christian Brunette [6, 76] (which also contains a contribution to our objective (12) on efficient implementations of the reactive model), where the definition and implementation of ICOBJS is described, with some applications.

The FAIRTHREADS are an instance of the reactive programming paradigm, where concurrency is introduced by spawning threads (as opposed to using a parallel composition construct). They were first implemented as a JAVA API. In the implementation in C (again as an API, on top of the Pthreads library) that has been achieved during the evaluation period [17, 73], the notion of a *scheduler* has been added, with features to create schedulers, and to link and unlink threads with respect to schedulers. A scheduler is a “reactive area”, where the threads (linked to this area) perform in a cooperative manner, sharing instants and broadcast signals. The various reactive areas that are active at some point are executed in an asynchronous manner, with a preemptive scheduling. This model fulfills the purposes we identified with our objective (4), and also in (2) (link/unlink constructs). The combination of cooperative (inside a scheduler) and preemptive (among schedulers) modes of execution allows the programmer to unlink a thread when it has to perform a blocking I/O for instance. This model also opens a way to true concurrency, where

asynchronous threads are executed on different processors. The API then evolved into the LOFT language [67, 74], and an alternative implementation, not using Posix Threads, was developed, suited for embedded systems with limited resources. The latter has been used for a prey-predator demo running on the Nintendo GBA platform (see [74]), as announced in our objective (5). LOFT has also been used to program in an efficient and modular way cellular automata [68, 75], where the cells are simple fair threads, created on demand. These experiments clearly show the efficiency of the reactive model, both at the design and execution stage. However, “reactive programming in C”, as it is implemented with LOFT, sounds like an antinomy, since the reactive model is supposed to have a clear, formal semantics. For instance, if dereferencing a null pointer is a possibility, one may wonder what the notion of an instant is. Then we made an attempt at developing an alternative implementation of LOFT, based on CYCLONE [69], with the objective of building a safe concurrent language on top of it. However, to this purpose one should in particular develop some analyses in order to maintain the *reactivity* property, that is the fact that each instant in the execution actually terminates, and it was difficult to see how to address such an issue using CYCLONE as the underlying language. We shall come back to this question in the section on Security, but a lesson is that one should better use a programming language supporting formal analyses as a basis for an integrated “safe” reactive programming language. (In the MIMOSA team, our choice in this respect is in functional – and imperative – languages, such as SCHEME or ML.) As a preliminary step towards “safe” reactive and concurrent programming, we have addressed the issue of avoiding data races in a (first-order, functional and reactive) core language, where threads can be unlinked from schedulers, and run in a preemptive mode. More precisely, in [41] we designed a type and effect system that ensures that, when running in preemptive mode, threads do not interfere with other threads. We should point out that this direction of work, towards a safe concurrent (possibly running on a multi-processor architecture) and reactive model is new, and was not anticipated in our previous report.

Another line of work on the reactive model, which was not anticipated in 2003<sup>(2)</sup>, has been initiated during the evaluated period. It consists in a deep theoretical study of the model [12, 57, 58, 59], motivated by our work on the security issues raised in this model (see the section below on Security). We have started this work on the security aspects using a simplified version of the reactive model, and this gave rise to the question of *expressiveness* of the fragment we considered. In [57] we have shown that the reactive model can indeed be encoded, by means of a continuation-passing-style transformation, into the fragment without sequential composition, and where the only synchronization construct is the test for presence of a signal (with an “else” branch deferred to the next instant). Surprisingly (after many years of discussions about the primitives of the ESTEREL language), this is perhaps the first formal result regarding the expressivity of the reactive constructs. The criterion used for establishing the soundness of the translation is the equality of traces, but this is not a very manageable notion of program equivalence. In [57] we therefore also study the theory of program equivalence in the reactive model, and our main result is the characterization of program equivalence by means of a “labelled bisimulation”, which provides a better proof method than trace equivalence. This has been extended in [58], dealing with first-order data, including signals, that can be transmitted along signals. The resulting model is similar to the  $\pi$ -calculus, but with a reactive style as regards concurrency and communication. We show in [58] that the bisimulation equivalence previously introduced can be generalized, and that it characterizes contextual equivalence, which is the adaptation of the standard notion to the reactive model. This sets up solid foundations for further theoretical investigations in this model.

#### 2.4.4 External support

The works [12, 57, 58, 59] were supported by the ACI project CRISS, whereas [41] was supported by the ACI project ALIDECS.

#### 2.4.5 Self assessment

Our main achievements under the heading of “safe parallel and reactive programming” have been in the two unexpected lines of work, towards the safe integration of the reactive style of program-

---

<sup>(2)</sup> In [Mimoso03] we wrote however that we were taking “*some decisions, like for instance to try to orient the theoretical work towards the computing models that are developed within the project*” (page 18).

ming into the more classical one of preemptive multithreading, and towards a theoretical model of reactive programming. Work on the synchronous  $\pi$ -calculus in particular offers the first formal study of a notion of equivalence in the reactive model. We think that in both directions our work is still preliminary, and that it needs to be deepened. However, our work on the mixed cooperative/preemptive model, extending the reactive one with link/unlink constructs, should not be underestimated, since it might evolve into a concurrent programming model allowing one to take advantage of modern architectures. We think that experimenting this in multi-core programming is a way that deserves to be explored, where we would find justifications for our theoretical investigations of the model.

## 2.5 Mobile code and web programming languages

### 2.5.1 Personnel

G erard Boudol, Fr ed eric Boussinot, Yannis Bres, Silvano Dal Zilio, Xudong Guan, Damien Ciabrini, Erick Gallesio, Florian Loitsch, Ana Matos, Manuel Serrano, Pascal Zimmer.

### 2.5.2 Project-team positioning

The work that we did in the MIKADO project on global computing models is now continuing in the IST Global Computing II project SENSORIA, of which we are not a partner. This work is now oriented towards web services, a concept that we are also developing, but from a programming language perspective. This approach is shared by some other groups, which however concentrate on specific aspects, where we intend to build an integrated tool. Regarding XML processing, Benjamin Pierce (Penn University), Phil Wadler (Edinburgh) and others have proposed XDUCE, which was followed by CDUCE of Guiseppe Castagna (ENS Paris) and Alain Frisch (INRIA GALLIUM – formerly CRISTAL), and the W3C has proposed XQUERY. Facilities for programming web GUIs are found in open-source contributions like DOJO or OPENLASZLO, and in FLAPJAX by Krishnamurthi (Brown University), which is inspired by data-flow synchronous programming. Phil Wadler recently proposed LINKS, to deal with the distributed computing dimension of web applications. OCSIGEN by Vincent Balat (Paris 7 University) is another language suited for these applications. Finally, there is the Google’s proposal GWT (Google Web Toolkit), which offers an ad hoc development environment, hiding web peculiarities. In the community of functional (and imperative) programming, our team competes with PLT SCHEME, a joint effort of Northeastern, Brown, Chicago and Utah Universities oriented towards educational usages, and with OCAML (INRIA GALLIUM) or GHC (Microsoft Research).

### 2.5.3 Scientific achievements

At the beginning of the evaluated period we did some work on theoretical models of mobility, like the  $\pi$ -calculus [13, 51], and the calculus of Mobile Ambients [19, 24, 25, 47, 56]. The work we did in the IST MIKADO project on a “membrane calculus” [33, 60, 61, 63] is in the same vein; this work fulfils the objective (9) in [Mimosa03], which was about the notion of a “programmable membrane.” In the objective (11), we stated our intention to work on logical approaches for reasoning about the spatial structure of systems, which emerged from the application of concepts from the Ambients calculus to unstructured data (and in particular XML documents). Our work on this topic is reported in [26, 43, 44].

With our objective (10), we planned to investigate the use of the intersection type discipline as a possible alternative to ML polymorphism, in order to obtain an implicitly typed version of an extension of our mixins model (see [16, 62, 85]) to prototypes (the operational encoding of prototypes we had was not published). This was conceived as a preliminary step towards an extension of ULM (see below) into a more realistic language. Then we had some theoretical work on typing with intersection types [31, 37, 65], showing the strong normalization theorem in a new way, and introducing a new semi-algorithm for type inference, which was implemented by Pascal Zimmer [86]. Some preliminary results regarding the extension to a richer language, including references and recursion, as needed for the encoding of mixins and prototypes, were obtained in Zimmer’s PhD Thesis [10], but the ultimate goal of applying this to the typing of prototypes was

not reached. Since then, this line of research has not been pursued (partly because Zimmer left the group for a job at Wall Street).

We made several contributions to the development of our SCHEME implementation. In particular, we have developed BUGLOO [7, 20, 39, 40, 77], a source-level debugger for SCHEME programs compiled with BIGLOO into JVM byte-code. Besides the classical features found in debuggers, BUGLOO also offers trace manipulations, debug sessions and memory debugging. As announced with our objective (2), this has been extended to the language enriched with FAIRTHREADS (see below), showing the superiority of the reactive approach over classical multi-threading, as regards the development and maintenance of concurrent programs. Another contribution has been the design and implementation of the SKRIBE language [22, 80]. SKRIBE is designed for authoring documents, such as web pages, technical reports, or API documentations, and it allows parts of the document to be produced in a dynamic way. The implementation of SKRIBE is now a stable software, which is used in our team to produce various documents, such as our web page. Also relevant to this line of work on functional programming are [18, 21, 38, 46, 48]. Some further work on BIGLOO and SKRIBE, which is not reported upon in technical publications, is described in our annual activity reports. However, we must say that we did not tackle our objective (3), about static optimizations regarding SCHEME FAIRTHREADS.

The work consisting in integrating Boussinot's FAIRTHREADS (see Section 2.4.3) in Serrano's implementation of SCHEME was partly done during the previous evaluation period. This work has been finalized, and is now published in [53]. In SCHEME FAIRTHREADS, the "unlinking" construct is not directly available to the programmer, and is replaced by the notion of *service threads*, offering support for asynchronous interactions with the underlying operating system. With our objective (1), we planned to further enrich this with programming constructs for mobile code, by first designing a semantical model of migration in the reactive and functional approach. This has been done in [32], where we argue that the reactive style, allowing a program to react to the presence or absence of particular events, and to behave according to the time it takes to perform some computations, is well suited for programming agents roaming in a global context, having to deal with various kinds of failures, such as unpredictable delays, transient disconnections, or traffic congestion. The resulting core language, called ULM (for "Un Langage pour la Mobilité"), has been integrated into SCHEME, and implemented by means of a virtual machine [45, 78]. An alternative virtual machine for ULM has been designed, that is suited for embedded devices supporting J2ME (JAVA 2 Mobile Edition), such as most mobile phones. This contributes to our goal (5) on the implementation of reactive programs in embedded systems. We are now beginning to use this implementation in an application demonstrating the use of mobile code for dynamic service reconfiguration (this is partly supported by a new grant from France Télécom R&D).

Our most recent effort in the area of programming languages, which was not anticipated in 2003, is with the HOP language for programming web applications [52, 54, 83]. In order to cover the wide variety of tasks involved in these applications (for accessing and processing information from databases, and interacting with the user), traditional web developments use a plethora of "domain specific languages" which are known as *web technologies*. The only help one gets in front of this new Babel tower is the normalization effort supported by the World Wide Web consortium, known as W3C *recommendations*. However, taming many different languages is still required even for simple applications. In order to remove this burden from programmers we are proposing a new programming language, especially tuned for the web, the main characteristic of which is to address all the aspects of a web application inside a single formalism. Not surprisingly, HOP is built on top of SCHEME, which is particularly convenient for manipulating tree-like structures. The basis of the HOP programming model is the "web broker", which is a programmable, multi-threaded web server and proxy, in charge of all the computations that involve the resources of the clients (in cooperation with web browsers for the GUI aspects), and of the communications with other HOP brokers, or regular web servers. Managing distributed computations in HOP is achieved using a specific, asynchronous form of remote service invocation, but we are now considering the use of mobile code. HOP programs are compiled into JAVASCRIPT, which is the language used in web browsers, by means of the SCHEME2JS compiler [81]. A first version of HOP has been released in May 2006 [83].

#### 2.5.4 Collaborations

B. Serpette from the OASIS team at INRIA Sophia Antipolis was a close collaborator on functional programming. He co-authored [18, 38, 53].

#### 2.5.5 External support

The works [33, 47, 51, 60, 61, 63] were supported by the IST Global Computing MIKADO project. The work [32] was supported by MIKADO and the ACI CRISS projects, and [45, 78] were supported by CRISS. The works [18, 38] were supported by the Microsoft Rotor grant.

#### 2.5.6 Self assessment

Our two main achievements here are the design and implementation of ULM, and of the HOP language. The development of ULM was not driven by applications, but proceeded from formal semantical models that we previously studied. From this point of view, ULM is perhaps the first (core) language for mobile code having a mathematical semantics<sup>(3)</sup>. With ULM we are now reaching the point where this language is used in a telecom application. HOP, on the contrary, was built from the desire of having a convenient language for programming web 2.0 applications. From this point of view, we think this language is much more mature (and usable) than any of its competitors. We hope that the two efforts will meet in a near future: it would be interesting to provide HOP with a formal semantics, and to integrate mobile code facilities in this language.

## 2.6 Security

### 2.6.1 Personnel

Roberto Amadio, Gérard Boudol, Ilaria Castellani, Frédéric Dabrowski, Ana Matos, Vincent Vanackère.

### 2.6.2 Project-team positioning

Security is an extremely active research area in Computer Science, and there are by now numerous journals, conferences, and workshops devoted to topics relevant to security. Our work pertains to the language-based approach to security, which again is a very active area, especially regarding secure information flow (let us mention the groups of Andrew Myers at Cornell University, Steve Zdancewic at University of Pennsylvania, Andrei Sabelfeld and Dave Sands at Chalmers, of David Naumann at Stevens Institute of Technology and Anindya Banerjee at Kansas State University, to name just a few). Controlling code complexity is usually not classified as a security issue. The methods we use on this topic are close to the ones used in studying rewriting systems (with a tremendous amount of work devoted to termination), especially by Jean-Yves Marion (CALLIGRAMME team of INRIA's SymC), but we also collaborated with people working on functional languages with a logical, or a typing approach (initiated by Jean-Yves Girard and Yves Lafont, with contributions by Martin Hofmann in the IST Global Computing "Mobile Resources Guarantees" project for instance), and especially with Patrick Baillot. At INRIA the team that is the closest to MIMOSA as regards security is EVEREST, but some other teams are also dealing with security issues, mainly in the SymA Theme (CASSIS, LANDE, SECSI, and MOSCOVA), and many others, again mainly in SymA, investigate safety issues. Our ultimate goal in security is to provide the programmer who has to develop security-minded software with tools that could help avoiding errors at the design stage. Then, although we had some work on byte-code verification, we are more inclined towards static analysis techniques, and especially type systems.

### 2.6.3 Scientific achievements

In the security area we had three main objectives, numbered (6), (7) and (8) in [Mimosa03]. Our objective (7) was about the logical specification of cryptographic protocols, with two goals: decision methods and comparison with other models. The first goal has been achieved in the

---

<sup>(3)</sup> To our view the PICT or JOIN languages are not really dealing with mobile code, but rather with mobile links.

implementation and optimization of the TRUST software [9, 55, 84]. Apart from that we did not pursue the work on cryptographic protocols, which is actively investigated in other groups, and is not so closely related to mobile code.

The objective (6) was about secure information flow. We recall that the overall objective here is to design methods complementing traditional access control, in order to achieve end-to-end information confidentiality, by checking that programs do not implement illegal information flow from confidential information to public one. In the “language-based security” approach, checking secure information flow is usually achieved by means of dedicated type systems, and our objective was to design such a type system for the ULM language, and to prove its soundness. As noted in [Mimosa03], this task involves defining a “non-interference property” for reactive systems, since these rely on a semantical model which is quite different from the sequential one (in particular, new kinds of information leak arise in this model). In [50, 70], we defined such a property, we designed a security type system for a core language for reactive programming, and proved its soundness. However, the language we considered is far less expressive than ULM: it is a combination of a small imperative “while-language” with the reactive constructs. Then we had to extend this to the functional and mobility aspects of ULM. This is what we did, in the context of an important issue in secure information flow, that in 2003 we did not expect to tackle, namely the one of *declassification*. Indeed, there are many examples of useful programs that perform some declassification, and these are rejected as insecure according to the non-interference criterion, which therefore appears to be of little use in practice. Then a question is: how to enrich a programming language with declassification mechanisms, in such a way that some security property could still be shown to hold? This is the question we addressed in [36], where we introduced a construct for declaring local flow policies, and a corresponding “local non-interference” property. We also introduced a security type system for a quite expressive language, involving imperative and higher-order (functional) constructs, as in ML (and ULM). We followed the classical types-and-effects style, providing a “state-oriented” approach to information flow security, that differs from the more conventional (but less intuitive, to our view, and less in line with classical access control) “value-oriented” approach followed in functional languages (like FlowCAML for instance). In [34] the type system was further refined, in order to get a typing which is as precise as possible while ensuring the required security property. The journal paper [66] improves on [36] by incorporating the refined analysis of [34]. The code mobility features of ULM, which introduce new kinds of information leak, were investigated in [49]. Most of our results on information flow security in this period can be found in Ana Matos’ PhD Thesis [8].

The objective (8) of [Mimosa03] was about controlling the use of computing resources: the goal here is to design methods – hopefully effective static analyses – to ensure that a program makes a reasonable use of time and space for instance, that is, the goal is to control the computational *complexity* of code. One can perform some control using dynamic checks, but in some cases – embedded systems or mobile code for instance –, one would prefer to have ways of rejecting “dangerous” code without having to run it. We noticed in [Mimosa03] that in reactive programming there is a related issue: one has to ensure that every instant terminates (and preferably in short time), that is, one has to ensure the *reactivity* of a program. Again, the original goal was to find means to ensure “feasible reactivity” for the ULM language, but it was noted that, given the expressiveness of the language, this is a very ambitious goal, and that even more modest objectives were already very challenging. Our work started with [11, 27], showing that one can use polynomial interpretations of rewrite systems in the max-plus algebra in order to characterize polynomial complexity classes, and that these interpretations can be synthesized, thus opening the way for static analysis techniques. We then used (quasi-)interpretations, together with standard techniques for showing termination, in byte-code verification of first-order functional programs [15, 29, 42, 72], thus certifying polynomial bounds for these programs. As a step towards “feasible reactivity”, we extended this work to the setting of synchronous, cooperative threads [30]. In this work we show that, if its functional part admits a polynomial quasi-interpretation, then a synchronous program runs in space polynomial in the size of its parameters at the beginning of the instant. However, we left open the problem of guaranteeing that the parameters of the program stay within certain bounds, and thus that the resources needed for execution are controlled over arbitrarily many instants. This was solved in [14, 28], by requiring that the computations of the system at a given instant do not depend on the values that have been read during previous instants.

### 2.6.4 External support

All the works mentioned above except [9, 55] were supported by the ACI CRISS project.

### 2.6.5 Self assessment

Regarding secure information flow, our main achievements are: 1. the definition of a security policy that takes declassification into account (while still rejecting programs that contain “security errors”), something that had long been searched for, and 2. the design of a precise type system for a high-level, imperative and functional language, close to a realistic programming language. Regarding resource control, we have introduced the notion of “feasible reactivity,” and designed some analysis techniques to ensure this property in a first-order reactive language. This improves over what has been done for synchronous languages *à la* ESTEREL for instance, which simply ignores the actual computations done by a program, by assuming that these are “instantaneously” performed in an outer language. A challenge for the future is to make our work closer to applications, and more specifically to the web applications that begin to be tackled in our group. However, the goal of managing “feasible reactivity” by means of static analysis in the context of a realistic programming language seems very difficult, and more foundational work is required before we can hope solving this problem in a useful way. A step can be to address first the problem of ensuring reactivity in such a language.

## 3 Knowledge dissemination

### 3.0.6 Publications

	2003	2004	2005	2006	Total
PhD Thesis	1	3		2	6
H.D.R (*)					
Journal	4	3	2	5	14
Conference proceedings (**)	7	10	10	4	31
Book chapter	1				1
Book (written)					
Book (edited)	1		2	1	4
Patent					
Technical report		3	5	3	11
Deliverable	2		1		3
Total	16	19	20	15	70

(\*) HDR Habilitation à diriger des Recherches

(\*\*) Conference with a program committee

*Indicate the major journals in the field and, for each, indicate the number of papers coauthored by members of the project-team that have been accepted during the evaluation period.*

1. TOPLAS 1
2. J. of Functional Programming 3
3. Information and Computation 0
4. Mathematical Structures in Computer Science 1
5. Concurrency: Practice and Experience 1
6. Theoretical Computer Science 2

Some of our technical reports have been accepted for publication in journals, namely: [58] in Information and Computation, [65] in Theoretical Computer Science, [66] in the Journal of Computer Security, and [57, 70] in the Journal of Logic and Algebraic Programming.

*Indicate the major conferences in the field and, for each, indicate the number of papers coauthored by members of the project-team that have been accepted during the evaluation period.*

1. POPL 1
2. ICFP 0
3. CSFW 1
4. ESOP 1
5. CONCUR 1
6. TLCA 2

### 3.1 Software

Some of the softwares that we produced during the evaluated period are no longer maintained (mainly because their authors left the team), but are still freely available [71, 76, 77, 85, 84, 86]. Some others were built in a “proof-of-concept” perspective, and they are mainly used for our own experiments. These software are also freely available, together with some documentation and demos [72, 73, 74, 75, 78, 79].

- BIGLOO [82] is an optimizing compiler for the SCHEME programming language. It aims at delivering fast executables. It is provided with three different code generators, so that a same source code can either be compiled to native code for maximum performance, or to JVM or .NET byte codes for portability. From January 2003 to December 2006 we have released ten major versions of BIGLOO, under GPL license. Each of these versions has been announced on the regular open source community channels – news groups, freshmeat<sup>(4)</sup>. BIGLOO is shipped with most major Linux distributions (Debian, Ubuntu, Gentoo, Fedora, ...). BIGLOO has a small but steady community of users. Each year, around 1000 mails are sent to the mailing list. Some raise issues and problems. Others propose extensions. Some require for new features. BIGLOO is most frequently used in industrial contexts and for the development of open source softwares. In addition to being a vector of research (we use BIGLOO as a test-bed for new compilation techniques and for new programming constructions) we also use BIGLOO as a tool of choice for building other research projects. In particular, two of our other softwares, SKRIBE and HOP, are actually implemented in BIGLOO. BIGLOO is in direct competition with other SCHEME implementations. The most widely used are PLT-SCHEME<sup>(5)</sup> which is an integrated development environment specially tuned for teaching programming, and GUILLE<sup>(6)</sup> which at some point was used as the official GNU scripting language. In addition to these SCHEME implementations, BIGLOO also competes with implementations of other functional languages, most notably OCAML and HASKELL.
- SKRIBE [80] is a text processor, which looks like a mark-up language *à la* HTML. Even if it is a general purpose tool, SKRIBE best suits the writing of technical documents such as web pages or technical reports, API documentations, etc. SKRIBE’s application field slightly overlaps with HOP’s one. Both systems can be used for implementing web sites. However, SKRIBE is tuned for authoring static documents while HOP is tuned for dynamic documents and web applications. Since a couple of years we have focused our research interest on that last aspect that, we think, opens new research demands. Hence, we have reduced our implementation effort on SKRIBE to the benefit of HOP. SKRIBE’s version 1.2g was released on October 2006, under GPL license.
- HOP, a language for programming the web 2.0, has been one of our main research focus since the last two years. It is a higher-order language for programming interactive web applications such as web agendas, web galleries, music players, mashups, etc. HOP can be viewed as a replacement for traditional graphical toolkits. HOP implements the concept of a web broker, i.e. a server that may act indifferently as a regular web server or web proxy. HOP has been first publicly released in June 2006 along with a registration to the APP<sup>(7)</sup>. Its announcement has raised a strong interest demonstrated by the number of visits of its web site (more than 10.000 visitors have browsed

---

<sup>(4)</sup> <http://freshmeat.net/>

<sup>(5)</sup> <http://www.pltscheme.org>

<sup>(6)</sup> <http://www.gnu.org/software/guile/guile.html>

<sup>(7)</sup> No. IDDN.FR.001.260002.000.S.P.2006.000.10400.

it in the week following the announcement). HOP is in competition with languages designed for programming web applications without tiers. As such, its most important competitor is LINKS<sup>(8)</sup>, a research project headed by Phil Wadler from Edinburgh University. LINKS and HOP share the same ultimate goal but the two systems show different approaches and a different implementation investment. More precisely, LINKS is still highly experimental and it is not mature enough to be used for developing realistic applications. On the other hand, HOP is used on a daily basis. For instance, it is used for implementing the HOP web site, and it has been used since two years for implementing the INRIA Sophia repository of annual team research reports. HOP's version 1.5.0 was released on February 2007 under GPL license.

### 3.1.1 Valorization and technology transfert

The BIGLOO compiler is a vector of “valorization” for our team. This software is indeed used in academia by computer scientists (for instance it is used for teaching at the CNAM in Paris) and by researchers of other disciplines (for several years it has been extensively used by the scientific computation team of the Institut Pasteur) and it is also used in industry. For instance, as reported by an article published by Slashdot<sup>(9)</sup>, BIGLOO has been used by the Zend company for developing the first publicly available compiler for PHP5. BIGLOO is also frequently used by hardware makers such as Motorola, ST Microelectronics, etc.

BIGLOO can also occasionally play the role of a test-bed that we use with industrial partners. For instance, in collaboration with Texas Instruments, we have used it for experimenting with new compilation techniques needed by embedded devices, such as cell-phones, where optimizing for energy consumption could be more important than optimizing for pure velocity.

## 3.2 Teaching

- **Roberto Amadio** taught formal languages and compilers at a graduate level in 2002/2003 at the University of Provence, and he gave a DEA course on functional programming and resource control. In 2003/2004/2005 he was responsible for the Master of Computer Science at the University of Provence, still teaching at the graduate level. He gave a course on concurrency in the Master Parisien de Recherche en Informatique in 2005. He supervised the internship of Eric Levieil (ENS Paris) (2003) and J. Radhakrishnan (2004)
- **G rard Boudol** gave a PhD course on language-based security during the Security spring school at Marseilles in 2005. He supervised the internships of St phane Epardaud (with Manuel Serrano) (2004), Maoulida Daoudou (2005), and Marija Kolundzija (2006).
- **Fr d ric Boussinot** gave a DEA course on reactive programming at the University of Nice-Sophia Antipolis from 2002 to 2005. He supervised the internships of Olivier Para (2003) and Celio Troi (2005)
- **Silvano Dal Zilio** gave a course at the DEA in Computer Science of Marseilles Universities in 2003/2004. He supervised the internships of Etienne Duchesne (ENS Paris) (2003), Pierre-Paul Tacher (2004), and Yann Barsamian (ENS Lyon) (2005).
- **Manuel Serrano** gave a DEA course at the University of Nice-Sophia Antipolis from 2002 to 2005. He supervised the internships of Christian Loitsch (2003), Florian Loitsch and St phane Epardaud (with G rard Boudol) (2004)

MIMOSA is (or was) associated as an “ quipe d'accueil” with the DEA in Computer Science of Marseilles University, the Master STIC (Sciences et Technologies de l'Information et de la Communication) of the University of Nice-Sophia Antipolis, the Master of the  cole Normale Sup rieure de Lyon, and the “Master Parisien de Recherche en Informatique” of the  cole Normale Sup rieure de Paris, the  cole Polytechnique, and the University of Paris 7.

---

<sup>(8)</sup> <http://groups.inf.ed.ac.uk/links/>

<sup>(9)</sup> <http://developers.slashdot.org/article.pl?sid=04/07/13/2237233>

### 3.3 Visibility

During the evaluation period, members of the MIMOSA team were (the details are in our annual activity reports):

- **General chair** of CSFW'05.
- **PC chair** of: CONCUR'03, Formal Methods for Mobility'03 Workshop, EXPRESS'06.
- **PC members** of: FICS'03 Workshop, COORDINATION'04, 3rd IFIP Conf. on TCS, EXPRESS'04, ITRS'04, Foundations of Global and Ubiquitous Computing'04 Workshop, FST-TCS'04, WOOD'04 Workshop, ICFP'04, EXPRESS'05, FOSSACS'06 (two members), ICALP'06, CONCUR'06, Modélisation des Systèmes Réactifs'05 Workshop, SLAP'05 Workshop, ECOOP'05 Workshop on LISP and SCHEME, ICFP'05, MFCS'06, CSFW'06, ESOP'07.
- **Referees** for 13 PhD theses.
- **Examiners** for 4 PhD theses.
- **Referees** for 2 HDR.
- **Examiner** for 1 HDR.
- **Invited speakers** at: Italian Conf. on TCS'03, COORDINATION'04, Intern. Workshop on Security Analysis of Systems'04, DSL Workshop on Functional Programming and Verification'05, SCHEME and Functional Programming'05 Workshop, FST-TCS'06.

Moreover, R. Amadio is member of the steering committees of ETAPS and CONCUR, and of the French Spring School in TCS, and was editor of a special issue of the J. of Logic and Algebraic Programming on Modelling and Verification of Cryptographic Protocols; F. Boussinot is a member of the editorial board of TSI; M. Serrano was co-editor of two special issues of HOSC on SCHEME, is a member of the "SCHEME Strategy Group" that decides on the evolutions of this language, and has participated to the writing of the Revised 6 Report on the SCHEME programming language.

## 4 External Funding

(k euros)	2003	2004	2005	2006
<b>National initiatives</b>				
ACI CRISS	6	12	12	6
ACI GeoCal		1,5	1,3	
ACI ALIDECS			7,3	6,586
<b>European projects</b>				
IST MIKADO	98,192	98,192	11,5	
<b>Industrial contracts</b>				
Microsoft	15			
CTI France Télécom	22,87			
Texas Instruments				25
<b>Scholarships</b>				
PhD *	148,2	124,68	89,31	75,57
ODL#	7,5			
<b>Total</b>	<b>297,762</b>	<b>236,372</b>	<b>121,71</b>	<b>113,156</b>

\* other than those supported by one of the above projects

# engineer supported by INRIA

## National initiatives

- The ACI project **CRISS** (Resources and Interference Control in Synchronous Systems) started July 2003 and ended July 2006. It was coordinated by Roberto Amadio, and the participants were, besides MIMOSA and the Laboratoire d'Informatique Fondamentale of Marseilles, the LIPN from Paris-Villetaneuse (Patrick Baillot) and the INRIA project **CALLIGRAME** (Jean-Yves Marion) from LORIA in Nancy. The main aim was to study security issues raised by mobile code, in the setting of reactive programming (ULM).
- The ACI project **GeoCal** (Geometry of Computations) was led by Thomas Erhard (IML Marseilles and PPS Paris), for the period 2003-2006. It consisted of ten groups of researchers in logics, theoretical computer science and algebraic topology.
- The ACI project **ALIDECS**, coordinated by Marc Pouzet (LRI Paris-Sud University), started in 2004. The participants are, besides our team, the LIP6/LRI, Verimag Grenoble (Florence Maraninchi), Pop-Art INRIA Rhone-Alpes (Pascal Fradet), and the LaMI (Jean-Marc Delosme) Evry. The objective is to study an integrated development environment for the construction and use of safe embedded components.

## European projects

- The IST-FET Global Computing project **MIKADO** started January 2002 and ended April 2005. It was coordinated by Jean-Bernard Stefani (SARDES team at INRIA Rhone-Alpes), and our partners were France Télécom R&D Grenoble, and the universities of Sussex (Matthew Hennessy), Lisbon (Vasco Vasconcelos), Florence (Rocco De Nicola) and Turin (Mariangiola Dezani). The objectives were to study programming models for distributed and mobile applications, related specification and analysis formalisms, and to design relevant programming constructs and corresponding prototypical virtual machines.

## Industrial contracts

- The Rotor grant from Microsoft was devoted to the adaptation of the Bigloo compiler to the .NET platform. We added a new bytecode generator to the compiler, and extended the runtime system for integration into Microsoft's Common Language Runtime.
- The objective of the 3 years CTI contract "Migrating Objects" with France Télécom R&D was to develop formal techniques for modeling, specifying and verifying migrating objects.
- The objective of our work with Texas Instruments was to study efficient code generation on the JSM, a proprietary architecture. More specifically, we adapted the BIGLOO compiler for this architecture, measured the impact of code generation optimizations embedded in BIGLOO, and developed new optimizations based on optimistic register allocation.

## 5 Objectives for the next four years

One can see from the figures we presented above that there has been, during the evaluated period, some decrease in the MIMOSA project as regards funding, personnel (especially PhD students), and to some extent publication record. Then one of our immediate objectives is to straighten up this situation. As indicated above, some of our papers will appear this year in technical journals, and our research plans should quickly materialize into conference publications. Regarding the funding, we participated in the IST second call for projects on Global Computing, as partners of the ACTIWEB IP proposal, and the proposed Network of Excellence "Foundations of Global Computing". Unfortunately we failed, and the Global Computing community as a whole has been reduced to two accepted projects (SENSORIA and AEOLUS – not counting the MOBIUS project on security). However, two new contracts were secured very recently by MIMOSA. First, we got support from France Télécom R&D, under the form of a CRE (Externalized Research Contract). This grant started in April 2006, but the funding was deferred to our 2007 budget. The project will last for three years, and the total funding is 120 kEuros. The objective is the analysis of security properties in programming environments for global computing, and more specifically the analysis of code complexity and information flow for ULM programs. Second, in January 2007 started a

national initiative, coordinated by G. Boudol, on security and concurrency (PARSEC project, for “Parallélisme et Sécurité”, of the ANR<sup>(10)</sup> “Sécurité Informatique” programme). Our partners are the EVEREST, LANDE and MOSCOVA teams at INRIA, and the Concurrency Group at the PPS laboratory of Paris 7 University. This project will last for four years, and the total funding is 428 kEuros, plus a PhD grant. Finally, we are actively looking for PhD students, proposing thesis topics in the relevant Masters, PhD schools, forums and mailing lists, but we do not always succeed in finding students who would be both fascinated by the topic and having the right skills for the research work. We will keep on searching for students.

Before we discuss our future research directions, we have to announce that, most likely – although no firm decision has been taken yet –, Gérard Boudol will not be head of a research team for the next evaluation seminar. This means that the MIMOSA project is likely to be reorganized before 2011, with new main objectives. Nevertheless, we still have a research programme for the next few years. This is presented under the same headings as above.

• **Concurrent programming.** Here we have two main directions. One is concerned with *safety*. As we have already argued, there is a need for user-level multi-threading facilities. Classical, preemptive multi-threading is notoriously difficult, mainly because of data races, and therefore one has to either design and implement tools to ensure safe programming with preemptive scheduling, or to propose alternative concurrency semantics. We have started working on this, in our model of mixed cooperative/preemptive model, with techniques for statically checking that threads executing in preemptive mode do not introduce data races. This work will be extended to a richer setting. We also plan to find means to ensure the reactivity property, in a high-level language integrating functional, imperative and reactive programming (see [35]). Another way of achieving safety that we plan to explore is in using *deterministic* concurrent programming models. It is clear that, in many applications, observable determinacy is a desirable property, both at the specification level, and also at the implementation level, to facilitate testing, verification and maintenance of the software. We shall study this property in the framework of the synchronous  $\pi$ -calculus, and in the SHIM model recently proposed by Olivier Tardieu and Stephen Edwards. Regarding safety in concurrent programming, we are also interested in understanding memory models, and finding which concurrent programming constructs may be used in a safe way together with relaxed models. The other main direction regarding concurrency we plan to investigate is more pragmatic: we would like to find appropriate means to take advantage of *multi-core* architectures in (concurrent) application programming. This is already a hot topic, to which some workshops and conferences are devoted. We plan to experiment with, and possibly adapt, our mixed programming cooperative/preemptive model on multi-processor architectures.

• **Web programming.** In the area of web programming and mobile code, our main objective is to develop and promote the HOP programming language. Even though a first version already exists and is publicly available, this language is still incomplete and immature in many respects. For the first version of HOP, we have focused our efforts on the design and the implementation of multi-tiers execution. That is, a HOP program is composed of a single source text, it uses a single name space for functions and variables but it actually executes simultaneously on a server *and* a client. The connection between the two ends is automatically managed by the runtime system which, for instance, takes care of the marshaling required for cross-site function invocations. Then the language entirely rests on the support of the browser for dealing with connections and sessions. All this needs to be improved. Remote references could be a technical solution that we intend to study. Our current programming model is convenient for programming applications that do not require strong synchronizations. For instance, we have been able to implement successfully simple multi-players games that involve little communications between the different parties. However, we failed to implement in a sufficiently efficient way games that demand precise and frequent synchronizations (such as reactive video games). Even though we are aware that the success is uncertain, this is another area we plan to explore. More generally, we envision to investigate new applications. Nowadays almost all new applications have an open door to the web. Most of them are provided with a web interface. This brings new power to computers because an application can now be simultaneously executed by different users and from different physical locations. The web is the standard ubiquitous execution platform. In the past, hardware constraints have confined the web

---

<sup>(10)</sup> Agence Nationale de la Recherche, the french funding agency.

to personal full-fledged computers. Mobile phones, PDAs, MP3 players, digital cameras, etc., are now equipped with sufficient resources for being able to browse the web too. This will undoubtedly give birth to new applications that will demand support from programming languages. As we have suggested above, this support might feature mobile code facilities.

• **Security.** In this area, we have identified several objectives in the PARSEC proposal. One of these is to integrate standard access control mechanisms with flow policies. More specifically, we plan to design a uniform model featuring “stack inspection” mechanisms, as found in the JAVA Security Architecture, and our construct for dynamically declaring flow policies, in order to study the interactions between access control and secure information flow, including declassification. We will have to define a comprehensive security property, and to show that this can be ensured by the usual means (i.e. a type system). We would also like to “bridge the gap” between the usual non-interference property and the security type systems, which are known to reject many programs that are considered secure according to the non-interference criterion. Our intention is not to find more precise means to ensure non-interference, but, on the contrary, to find more appropriate statements (as a safety property, see [64]) of the security guarantees provided by type systems. That is, our point of view is that type systems provide the programmer with valuable tools to avoid (security) errors, and that we should better understand why programs are rejected by a security type system. Another direction of research is to compare the language-based approach to secure information flow with process calculi formalizations of the same problem. The two lines of research have developed to a large extent independently, and not much work has been devoted to relate them. In the same vein, we wish to pursue and extend our study of secure information flow in concurrent languages: as we said above, we have studied fragments of the ULM language from this point of view, but a full, integrated treatment is still to be achieved. Moreover, the new models for concurrent programming that we plan to investigate, as indicated above, also need to be studied from the security point of view. In particular, it is clear that web application programming, as supported with the HOP language, raises many security issues. Indeed, web 2.0 applications rely on code mobility: when a client visualizes an HTML page on his browser, along with an HTML tree he also downloads a program that is executed on his machine. Then simple solutions such as the “same domain” restriction that used to be adequate a couple of years ago when web pages were nearly static are no longer sufficient. We think that enforcing security of web applications requires solutions that embrace the whole spectrum of web programming. That is, we think that new security solutions must simultaneously cover the computations that take place on the servers and on the client and the communications between the two ends, and this needs support from the programming languages.

## References

- [1] R. Amadio and D. Lugiez (editors). Proceedings of the international conference on concurrency theory – CONCUR’03. Lecture Notes in Computer Science, 2761, Marseille, France, September 2003.
- [2] R. Amadio (editor). Journal of Logic and Algebraic Programming. Special issue on *Modelling and Verification of Cryptographic Protocols*, 64(2), 2005.
- [3] R. Amadio, J. Guttman and J. Herzog (editors). Proceedings 18<sup>th</sup> IEEE Computer Security Foundations Workshop. ISBN 0-7695-2340-4. Aix-en-Provence, 2005.
- [4] R. Amadio and I. Phillips (editors) Proceedings 13<sup>th</sup> International Workshop on Expressiveness in Concurrency. Technical Report 2006/10 Department of Computing, Imperial College, London. Revised version to appear in Electronic Notes in Theoretical Computer Science. Bonn, 2006.

## Doctoral dissertations

- [5] R. Acosta-Bermejo. *Rejo - Langage d’objets réactifs et d’agents*. Thèse de doctorat, École Nationale Supérieure des Mines de Paris, October 2003.

- [6] C. Brunette. *Construction et simulation graphiques de comportements: le modèle des Icobjs*. Thèse de doctorat, Université de Nice-Sophia Antipolis, October 2004.
- [7] D. Ciabrini. *Débogage symbolique multi-langages pour les plates-formes d'exécution généralistes*. Thèse de doctorat, Université de Nice Sophia Antipolis, October 2006.
- [8] A. Matos. *Typing Secure Information Flow: Declassification and Mobility*. Thèse de doctorat, École Nationale Supérieure des Mines de Paris, January 2006.
- [9] V. Vanackere. *TRUST: un système de vérification automatique de protocoles cryptographiques*. Thèse de doctorat, Université de Provence, 2004.
- [10] P. Zimmer. *Récursion généralisée et inférence de types avec intersection*. Thèse de doctorat, Université de Nice Sophia Antipolis, 2004.

### Articles in referred journals and book chapters

- [11] R. Amadio. Synthesis of max-plus quasi-interpretations. *Fundamenta Informaticae*, 65:29–60, 2005.
- [12] R. Amadio, G. Boudol, F. Boussinot, and I. Castellani. Reactive concurrent programming revisited. *Electronic Notes in Theoretical Computer Science*, 162:49–60, 2006.
- [13] R. Amadio, G. Boudol, and C. Lhoussaine. The distributed receptive  $\pi$ -calculus. *ACM Transactions of Programming Languages and Systems (TOPLAS)*, 25(5):549–577, 2003.
- [14] R. Amadio and F. Dabrowski. Feasible reactivity for synchronous cooperative threads. *Electronic Notes in Theoretical Computer Science*, 154-3, 2006.
- [15] R. Amadio and S. Dal Zilio. Resource control for synchronous cooperative threads. *Theoretical Computer Science*, 358:229–254, 2006.
- [16] G. Boudol. The recursive record semantics of objects revisited. *Journal of Functional Programming*, 14:263–315, 2004.
- [17] F. Boussinot. Fairthreads: mixing cooperative and preemptive threads in C. *Concurrency and Computation: Practice and Experience*, 18:445–469, 2006.
- [18] Y. Bres, B. Serpette, and M. Serrano. Bigloo.NET: compiling Scheme to .NET CLR. *Journal of Object Technology*, 3(9), October 2004.
- [19] W. Charatonik, S. Dal Zilio, A.D. Gordon, S. Mukhopadhyay, and J-M. Talbot. Model checking mobile ambients. *Theoretical Computer Science*, 308(1):277–331, November 2003.
- [20] D. Ciabrini. Stack virtualization for source level debugging. *To appear in Software: Practice and Experience*, 2006 (published on line November 2).
- [21] E. Gallesio and M. Serrano. Programming Graphical User Interfaces with Scheme. *Journal of Functional Programming*, 13(5):839–886, September 2003.
- [22] E. Gallesio and M. Serrano. Scribe: a Functional Authoring Language. *Journal of Functional Programming*, 2005.
- [23] M. Serrano. *Le Langage C++*, volume H 3 078 of the encyclopedia *Techniques de l'Ingénieur*, pages 1–19, 2003.
- [24] D. Teller, P. Zimmer, and D. Hirschhoff. Using ambients to control resources. *International Journal of Information Security*, 2:126–144, 2004.

- [25] P. Zimmer. On the expressiveness of pure safe ambients. *Mathematical Structures in Computer Science*, 13, 2003.

## Publications in Conferences and Workshops

- [26] L. Acciai, M. Boreale, and S. Dal Zilio. A Typed Calculus for Querying Distributed XML Documents. In *NWPT 2005 – 17th Nordic Workshop on Programming Theory*, October 2005.
- [27] R. Amadio. Max-plus quasi-interpretations. In *Proc. Typed Lambda Calculi and Applications (TLCA 2003)*, Lecture Notes in Computer Science 2701, Valencia, Spain, 2003.
- [28] R. Amadio and F. Dabrowski. Feasible reactivity for synchronous cooperative threads. In *12th workshop Expressiveness in Concurrency*, San Francisco, USA, August 2005.
- [29] R. M. Amadio, S. Coupet-Grimal, S. Dal Zilio, and L. Jakubiec. A Functional Scenario for Bytecode Verification of Resource Bounds. In *CSL 2004 – 18th International Conference on Computer Science Logic*, volume 3210 of *Lecture Notes in Computer Science*, pages 265–279. Springer-Verlag, 2004.
- [30] R. M. Amadio and S. Dal Zilio. Resource Control for Synchronous Cooperative Threads. In *CONCUR 2004 – 15th International Conference on Concurrency Theory*, volume 3170 of *Lecture Notes in Computer Science*, pages 68–82. Springer-Verlag, 2004.
- [31] G. Boudol. On strong normalization in the intersection type discipline. In *TLCA '03*, number 2701 in *Lecture Notes in Computer Science*, pages 60–74, 2003.
- [32] G. Boudol. ULM, a core programming model for global computing. In *ESOP'04*, volume 2986 of *Lecture Notes in Computer Science*, pages 234–248, 2004.
- [33] G. Boudol. A generic membrane model. In *Second Global Computing Workshop*, volume 3267 of *Lecture Notes in Computer Science*, pages 209–222, 2005.
- [34] G. Boudol. On typing information flow. In *International Colloquium on Theoretical Aspects of Computing*, volume 3722 of *Lecture Notes in Computer Science*, pages 366–380, 2005.
- [35] G. Boudol. Shared-variable concurrency: a proposal. In *FST-TCS'06*, volume 4337 of *Lecture Notes in Computer Science*, 2006.
- [36] G. Boudol and A. Matos. On declassification and the non-disclosure policy. In *Computer Security Foundation Workshop*, pages 226–240, 2005.
- [37] G. Boudol and P. Zimmer. On type inference in the intersection type discipline. *ITRS'04 Workshop, Electronic Notes in Theoretical Computer Science*, 136:23–42, 2005.
- [38] Y. Bres, B. Serpette, and M. Serrano. Compiling Scheme programs to .NET Common Intermediate Language. In *2nd International Workshop on .NET Technologies*, Plzen, Czech Republic, May 2004.
- [39] D. Ciabrini. Debugging scheme fair threads. In *Proceedings of the 5th ACM-SIGPLAN Workshop on Scheme and Functional Programming*, pages 75–88, September 2004.
- [40] D. Ciabrini and M. Serrano. Bugloo: A source level debugger for scheme programs compiled into JVM bytecode. In *3th International Lisp Conference*, New York, USA, October 2003.
- [41] F. Dabrowski and F. Boussinot. Cooperative threads and preemptive computations. In *Proceedings of TV'06, Multithreading in Hardware and Software: Formal Approaches to Design and Verification*, Seattle, 2006.
- [42] S. Dal Zilio and R. Gascon. Resource Bound Certification for a Tail-Recursive Virtual Machine. In *APLAS 2005 – 3rd Asian Symposium on Programming Languages and Systems*, volume 3780 of *Lecture Notes in Computer Science*, pages 247–263. Springer-Verlag, November 2005.

- [43] S. Dal Zilio and D. Lugiez. XML schema, tree logic and sheaves automata. In *RTA 2003 – 14th International Conference on Rewriting Techniques and Applications*, volume 2706 of *Lecture Notes in Computer Science*, pages 246–263. Springer, June 2003.
- [44] S. Dal Zilio and D. Lugiez. A logic you can count on. In *POPL 2004 – 31st Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, 2004.
- [45] S. Epardaud. Mobile reactive programming in ULM. In O. Shivers and O. Waddell, editors, *Proceedings of the Fifth ACM SIGPLAN Workshop on Scheme and Functional Programming*, pages 87–98, 2004.
- [46] E. Galesio and M. Serrano. Ubiquitous mail. In *Proceedings of the Sixth Workshop on Scheme and Function Programming*, pages 69–76, Tallinn, Estonia, September 2005.
- [47] X. Guan. Towards a tree of channels. In *Foundations of Global Computing*, Vladimiro Sassone, editor, *Electronic Notes in Theoretical Computer Science*, volume 85. Elsevier, 2003.
- [48] F. Loitsch. Javascript compilation. In *Proceedings of the Sixth Workshop on Scheme and Functional Programming*, pages 101–111, Tallinn, Estonia, Sep 2005.
- [49] A. Matos. Non-disclosure for distributed mobile code. In *FST-TCS'05*, volume 3821 of *Lecture Notes in Computer Science*, 2005.
- [50] A. Matos, G. Boudol, and I. Castellani. Typing noninterference for reactive programs. In *Foundations of Computer Security*, 2004.
- [51] A. Ravara, A. Matos, V.T. Vasconcelos, and L. Lopes. Lexically scoped distribution: what you see is what you get. In *Foundations of Global Computing*, Vladimiro Sassone, editor, *Electronic Notes in Theoretical Computer Science*, volume 85. Elsevier, 2003.
- [52] M. Serrano. The HOP Development Kit. Invited paper at the *Seventh ACM SIGPLAN Workshop on Scheme and Functional Programming*, Portland, Oregon, USA, September 2006.
- [53] M. Serrano, F. Boussinot, and B. Serpette. Scheme Fair Threads. In *6th ACM SIGPLAN International Conference on Principles and Practice of Declarative Programming (PPDP)*, pages 203–214, August 2004.
- [54] M. Serrano, E. Galesio, and F. Loitsch. HOP, a language for programming the Web 2.0. In *Proceedings of the First Dynamic Languages Symposium*, Portland, Oregon, USA, October 2006.
- [55] V. Vanackere. History-Dependent Scheduling for Cryptographic Processes. In *Proc. VMCAI' 2004*, *Lecture Notes in Computer Science*, Venice, Italy, 2004.
- [56] P. Zimmer. On the expressiveness of pure mobile ambients. In *EXPRESS'00*, L. Aceto and B. Victor, editors, *Electronic Notes in Theoretical Computer Science*, volume 39. Elsevier, 2003.

## Internal Reports

- [57] R. Amadio. The sl synchronous language, revisited. Technical report, Laboratoire PPS, Université de Paris 7, 2005. To appear in the *Journal of Logic and Algebraic Programming*, Vol. 70 (2007) 121-150.
- [58] R. Amadio. A synchronous  $\pi$ -calculus. Technical report, Université Paris 7, Laboratoire PPS, 2006, to appear in *Information and Computation*.
- [59] R. Amadio, G. Boudol, F. Boussinot, and I. Castellani. Reactive concurrent programming revisited. In *Algebraic Process Calculi: the first twenty five years and beyond*. Technical Report NS-05-3, BRICS Notes Series, 2005.

- [60] G. Boudol. Study of alternate distributed models, release 1. Deliverable D1.3.1, IST-FET Global Computing Project MIKADO, February 2003.
- [61] G. Boudol. A parametric model of migration and mobility, release 1. Deliverable D1.2.1, IST-FET Global Computing Project MIKADO, September 2003.
- [62] G. Boudol. Safe recursive boxes. Technical Report RR-5115, INRIA, February 2004.
- [63] G. Boudol. Study of alternate distributed models, release 2. Deliverable D1.3.2, IST-FET Global Computing Project MIKADO, March 2005.
- [64] G. Boudol. Secure information flow as a safety property. Unpublished draft, 2006.
- [65] G. Boudol. On strong normalization and type inference in the intersection type discipline. To appear in Theoret. Comput. Sci., 2006.
- [66] G. Boudol and A. Matos. On declassification and the non-disclosure policy. Revised version, to appear in the J. of Computer Security, 2005.
- [67] F. Boussinot. Concurrent Programming with Fair threads – The LOFT Language. <ftp://ftp-sop.inria.fr/mimosa/rp/LOFT/book.pdf>, 2004.
- [68] F. Boussinot. Reactive programming of cellular automata. Technical Report RR-5183, Inria, May 2004.
- [69] F. Boussinot. Cyclone+loft. Report 5680, INRIA, September 2005.
- [70] A. Matos, G. Boudol, and I. Castellani. Typing noninterference for reactive programs. Report 5594, INRIA, 2005, to appear in the Journal of Logic and Algebraic Programming (published on line February 26, 2007).

## Softwares

- [71] R. Acosta. REJO/ROS. <http://www-sop.inria.fr/mimosa/rp/ROS>
- [72] R. Amadio. SHAPE ANALYSIS. <http://www.pps.jussieu.fr/~amadio/Criss/shape-analysis.html>
- [73] F. Boussinot. FAIRTHREADS. <http://www-sop.inria.fr/mimosa/rp/FairThreads>
- [74] F. Boussinot. LOFT. <http://www-sop.inria.fr/mimosa/rp/LOFT>
- [75] F. Boussinot. CELLULAR AUTOMATA. <http://www-sop.inria.fr/mimosa/rp/CellularAutomata>
- [76] Ch. Brunette. ICOBJS [http://www-sop.inria.fr/mimosa/rp\\_2004/Icobjs](http://www-sop.inria.fr/mimosa/rp_2004/Icobjs)
- [77] D. Ciabrini. BUGLOO. <http://www-sop.inria.fr/mimosa/fp/Bugloo>
- [78] S. Epardaud. ULM. <http://www-sop.inria.fr/mimosa/Stephane.Epardaud/ulm>
- [79] S. Epardaud. LURC. <http://www-sop.inria.fr/mimosa/Stephane.Epardaud/lurc>
- [80] E. Gallesio and M. Serrano. SKRIBE. <http://www-sop.inria.fr/mimosa/fp/Skribe>
- [81] F. Loitsch. SCHEME2JS. <http://www-sop.inria.fr/mimosa/personnel/Florian.Loitsch/scheme2js>
- [82] M. Serrano. BIGLOO. <http://www-sop.inria.fr/mimosa/fp/Bigloo>
- [83] M. Serrano. HOP. <http://hop.inria.fr>
- [84] V. Vanackère. TRUST. <http://www.lif.univ-mrs.fr/~lugiez/Recherche/Logiciels/trust-1.1.tar>
- [85] P. Zimmer. MLOBJ. <http://www-sop.inria.fr/mimosa/Pascal.Zimmer/mlobj.html>
- [86] P. Zimmer. TYPI. <http://www-sop.inria.fr/mimosa/Pascal.Zimmer/typi.html>