INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Team indes

Informatique Diffuse et Sécurisée

Sophia Antipolis - Méditerranée

Theme : Distributed Systems and Services

Activity Report

2010

Table of contents

1. Team

Research Scientists

Gérard Berry [Research Director, Inria, HdR]
Gérard Boudol [Research Director, Inria, HdR]
Frédéric Boussinot [Research Director, CMA, HdR]
Ilaria Castellani [Research Scientist, Inria]
Tamara Rezk [Research Scientist, Inria]
Bernard Serpette [Research Scientist, Inria]
Manuel Serrano [Team Leader, Research Director, Inria, HdR]

Faculty Member

Christian Queinnec [Professor, University Pierre et Marie Curie - Paris 6, HdR]

Technical Staff

Marcos Dione [Inria]

PhD Students

Pejman Attar [Inria]
Johan Grande [Inria]
Zhengqin Luo [MENRT]
Cyprien Nicolas [Inria]
Gustavo Petri [IP Mobius]

Post-Doctoral Fellow

Thomas Gazagnaire [Inria, since February 28]

Administrative Assistant

Anais Cassino [Inria]

2. Overall Objectives

2.1. Overall Objectives

The goal of the Indes team is to study models for diffuse computing and develop languages for secure diffuse applications. Diffuse applications, of which Web 2.0 applications are a notable example, are the new applications emerging from the convergence of broad network accessibility, rich personal digital environment, and vast sources of information. Strong security guarantees are required for these applications, which intrinsically rely on sharing private information over networks of mutually distrustful nodes connected by unreliable media.

Diffuse computing requires an original combination of nearly all previous computing paradigms, ranging from classical sequential computing to parallel and concurrent computing in both their synchronous / reactive and asynchronous variants. It also benefits from the recent advances in mobile computing, since devices involved in diffuse applications are often mobile or portable.

The Indes team contributes to the whole chain of research on models and languages for diffuse computing, going from the study of foundational models and formal semantics to the design and implementation of new languages to be put to work on concrete applications. Emphasis is placed on correct-by-construction mechanisms to guarantee correct, efficient and secure implementation of high-level programs. The research is partly inspired by and built around Hop, the web programming model proposed by the former Mimosa team, which takes the web as its execution platform and targets interactive and multimedia applications.

3. Scientific Foundations

3.1. Parallelism, concurrency, and distribution

Concurrency magagement is at the heart of diffuse programming. Since the execution platforms are highly heterogeneous, many different concurrency principles and models may involved. Asynchronous concurrency is the basis of shared-memory process handling within multiprocessor or multicore computers, of direct or fifo-based message passing in distributed networks, and of fifo- or interrupt-based event handling in web-based human-machine interaction or sensor handling. Synchronous or quasi-synchronous concurrency is the basis of signal processing, of real-time control, and of safety-critical information acquisition and display. Interfacing existing devices based on these different concurrency principles within HOP or other diffuse programming languages will require better understanding of the underlying concurrency models and of the way they can nicely cooperate, a currently ill-resolved problem.

3.2. Web and functional programming

We are studying new paradigms for programming Web applications that rely on multi-tier functional programming [4]. This research has been initiated in the MIMOSA team, the predecessor of the INDES team, where we have created the new Web programming environment named HOP. It relies on a single formalism for programming the server-side and the client-side of the applications as well as for configuring the execution engine.

HOP is a functional language based on the SCHEME programming language. That is, it is a strict functional language, fully polymorphic, supporting side effects, and dynamically type-checked. HOP is implemented as an extension of the BIGLOO compiler that we develop [5]. In the past, we have extensively studied static analyses (type systems and inference, abstract interpretations, as well as classical compiler optimizations) to improve the efficiency of compilation in both space and time.

3.3. Security of diffuse programs

The main goal of our security research is to provide scalable and rigorous language-based techniques that can be integrated into multi-tier compilers to enforce the security of diffuse programs. Research on language-based security has been carried on before in both the MIMOSA [2] and EVEREST [1] teams. In particular previous research has focused on controlling information flow to ensure confidentiality.

Typical language-based solutions to these problems are founded on static analysis, logics, provable cryptography, and compilers that generate correct code by construction [3]. Relying on the multi-tier programming language HOP that tames the complexity of writing and analysing secure diffuse applications, we are studying language-based solutions to prominent web security problems such as code injection and cross-site scripting, to name a few.

4. Application Domains

4.1. Web programming

Along with games, multimedia applications, electronic commerce, and email, the web has popularized computers in everybody's life. The revolution is engaged and we may be at the dawn of a new era of computing where the web is a central element. The web constitutes an infrastructure more versatile, polymorphic, and open, in other words, more powerful, than any dedicated network previously invented. For this very reason, it is likely than most of the computer programs we will write in the future, for professional purposes as well as for our own needs, will extensively rely on the web.

In addition to allowing reactive and graphically pleasing interfaces, web applications are de facto distributed. Implementing an application with a web interface makes it instantly open to the world and accessible from much more than one computer. The web also partially solves the problem of platform compatibility because it physically separates the rendering engine from the computation engine. Therefore, the client does not have to make assumptions on the server hardware configuration, and vice versa. Lastly, HTML is highly durable. While traditional graphical toolkits evolve continuously, making existing interfaces obsolete and breaking backward compatibility, modern web browsers that render on the edge web pages are still able to correctly display the web pages of the early 1990's.
For these reasons, the web is arguably ready to escape the beaten track of n-tier applications, CGI scripting and interaction based on HTML forms. However, we think that it still lacks programming abstractions that minimize the overwhelming amount of technologies that need to be mastered when web programming is involved. Our experience on reactive and functional programming is used for bridging this gap.

4.2. Multimedia

Electronic equipments are less and less expensive and more and more widely spread out. Nowadays, in industrial countries, computers are almost as popular as TV sets. Today, almost everybody owns a mobile phone. Many are equipped with a GPS or a PDA. Modem, routers, NASes and other network appliances are also commonly used, although they are sometimes sealed under proprietary packaging such as the Livebox or the Freebox. Most of us evolve in an electronic environment which is rich but which is also populated with mostly isolated devices.
The first multimedia applications on the web have appeared with the Web 2.0. The most famous ones are Flickr, YouTube, or Deezer. All these applications rely on the same principle: they allow roaming users to access the various multimedia resources available all over the Internet via their web browser. The convergence between our new electronic environment and the multimedia facilities offered by the web will allow engineers to create new applications. However, since these applications are complex to implement this will not happen until appropriate languages and tools are available. In the Indes team, we develop compilers, systems, and libraries that address this problem.

4.3. House Automation

The web is the de facto standard of communication for heterogeneous devices. The number of devices able to access the web is permanently increasing. Nowadays, even our mobile phones can access the web. Tomorrow it could even be the turn of our wristwatches! The web hence constitutes a compelling architecture for developing applications relying on the "ambient" computing facilities. However, since current programming languages do not allow us to develop easily these applications, ambient computing is currently based on ad-hoc solutions. Programming ambient computing via the web is still to be explored. The tools developed in the Indes team allow us to build prototypes of a web-based home automation platform. For instance, we experiment with controlling heaters, air-conditioners, and electronic shutters with our mobile phones using web GUIs.

5. Software

5.1. Introduction

Most other software packages can be downloaded from the INDES

Web site:
http://www-sop.inria.fr/teams/indes

5.2. Functional programming

Participants: Frédéric Boussinot, Zhengqin Luo, Marcos Dione, Thomas Gazagnaire, Cyprien Nicolas, Tamara Rezk, Bernard Serpette, Manuel Serrano.

5.2.1. The Bigloo compiler

The programming environment for the Bigloo compiler [5] is available on the INRIA Web site at the following URL: http://www-sop.inria.fr/teams/indes/fp/Bigloo. The distribution contains an optimizing compiler that delivers native code, JVM bytecode, and .NET CLR bytecode. It contains a debugger, a profiler, and various Bigloo development tools. The distribution also contains several user libraries that enable the implementation of realistic applications.
BIGLOO was initially designed for implementing compact stand-alone applications under Unix. Nowadays, it runs harmoniously under Linux and MacOSX. The effort initiated in 2002 for porting it to Microsoft Windows is pursued by external contributors. In addition to the native back-ends, the BIGLOO JVM back-end has enabled a new set of applications: Web services, Web browser plug-ins, cross platform development, etc. The new BIGLOO .NET CLR back-end that is fully operational since release 2.6e enables a smooth integration of Bigloo programs under the Microsoft .NET environment.

5.2.2. Camloo

Camloo (http://www-sop.inria.fr/members/Thomas.Gazagnaire/) is a caml-light to bigloo compiler, which was developed few years ago to target bigloo 1.6c. New major releases 0.4.x of camloo have been done to support bigloo 3.4 and bigloo 3.5. Camloo make it possible for the user to develop seamlessly a multi-language project, where some files are written in caml-light, in C, and in bigloo. Unlike the previous versions of camloo, 0.4.x versions do not need a modified bigloo compiler to obtain good performance. Currently, the only supported backend for camloo is bigloo/C. We are currently rewriting the runtime of camloo in bigloo to get more portability and to be able to use HOP and camloo together.

5.2.3. CFlow

The prototype compiler "CFlow" takes as input code annotated with information flow security labels for integrity and confidentiality and compiles to F# code that implements cryptography and protocols that satisfy the given security specification.
Cflow has been coded in F#, developed mainly on Linux using mono (as a substitute to .NET), and partially tested under Windows (relying on .NET and Cygwin). The code is distributed under the terms of the CeCILL-B license (http://www.msr-inria.inria.fr/projects/sec/cflow/index.html).

5.2.4. The FunLoft language

FunLoft is a new programming language in which the focus is put on safety and multicore.
FunLoft is built on the model of FairThreads which makes concurrent programming simpler than usual preemptive-based techniques by providing a framework with a clear and sound semantics. FunLoft is designed with the following objectives:

- provide a safe language, in which, for example, data-races are impossible.
- control the use of resources (CPU and memory), for example, memory leaks cannot occur in FunLoft programs, which always react in finite time.
- have an efficient implementation which can deal with large numbers of concurrent components.
- benefit from the real parallelism offered by multicore machines.

A first experimental version of the compiler is available on the Reactive Programming. Several benchmarks are given, including cellular automata and simulation of colliding particles.

5.3. Web programming

Participants: Marcos Dione, Cyprien Nicolas, Manuel Serrano.

5.3.1. The HOP web programming environment

HOP is a new higher-order language designed for programming interactive web applications such as web agendas, web galleries, music players, etc. It exposes a programming model based on two computation levels. The first one is in charge of executing the logic of an application while the second one is in charge of executing the graphical user interface. HOP separates the logic and the graphical user interface but it packages them together and it supports strong collaboration between the two engines. The two execution flows communicate through function calls and event loops. Both ends can initiate communications.

The HOP programming environment consists in a web *broker* that intuitively combines in a single architecture a web server and a web proxy. The broker embeds a HOP interpreter for executing server-side code and a HOP client-side compiler for generating the code that will get executed by the client.

An important effort is devoted to providing HOP with a realistic and efficient implementation. The HOP implementation is *validated* against web applications that are used on a daily-basis. In particular, we have developed HOP applications for authoring and projecting slides, editing calendars, reading RSS streams, or managing blogs.

HOP has won the software *open source contest* organized by the ACM Multimedia Conference 2007 (http://mmc36.informatik.uni-augsburg.de/acmmm2007/). It is released under the GPL license. It is available at http://hop.inria.fr.

5.4. Old software

5.4.1. Skribe

SKRIBE is a functional programming language designed for authoring documents, such as Web pages or technical reports. It is built on top of the SCHEME programming language. Its concrete syntax is simple and looks familiar to anyone used to markup languages. Authoring a document with SKRIBE is as simple as with HTML or LaTeX. It is even possible to use it without noticing that it is a programming language because of the conciseness of its original syntax: the ratio *markup/text* is smaller than with the other markup systems we have tested.

Executing a SKRIBE program with a SKRIBE evaluator produces a target document. It can be HTML files for Web browsers, a LaTeX file for high-quality printed documents, or a set of *info* pages for on-line documentation.

5.4.2. Scheme2JS

Scm2JS is a Scheme to JavaScript compiler distributed under the GPL license. Even though much effort has been spent on being as close as possible to R5RS, we concentrated mainly on efficiency and interoperability. Usually Scm2JS produces JavaScript code that is comparable (in speed) to hand-written code. In order to achieve this performance, Scm2JS is not completely R5RS compliant. In particular it lacks exact numbers.

Interoperability with existing JavaScript code is ensured by a JavaScript-like dot-notation to access JavaScript objects and by a flexible symbol-resolution implementation.

Scm2JS is used on a daily basis within HOP, where it generates the code which is sent to the clients (web-browsers).

Scm2JS can be found here (http://www-sop.inria.fr/indes/scheme2js/)

5.4.3. ULM

ULM (*Un langage pour la mobilité*) is a language for mobility that was developed in the MIMOSA team. The ULM Scheme implementation is an embedding of the ULM primitives in the Scheme language. The bytecode compiler is available on PCs only but there are two ULM Virtual Machines: one for PCs and one for embedded devices supporting Java 2 Mobile Edition (J2ME) such as most mobile phones. The current version has preliminary support for a mixin object model, mobility over TCP/IP or Bluetooth Serial Line, reactive event loops, and native procedure calls with virtual machine reentry. The current version is available at http://www-sop.inria.fr/teams/mimosa/Stephane.Epardaud/ulm.

6. New Results

6.1. Security

Participants: Gérard Boudol, Ilaria Castellani, Zhengqin Luo, Tamara Rezk.

6.1.1. Security of Multithreaded Programs by Compilation

Information security is a pressing challenge for mobile code technologies. In order to claim end-to-end security of mobile code, it is necessary to establish that the code neither intentionally nor accidentally propagates sensitive information to an adversary. Although mobile code is commonly multithreaded low-level code, the literature is lacking enforcement mechanisms that ensure information security for such programs. This work offers a modular solution to the security of multithreaded programs. The modularity is three-fold: we give modular extensions of sequential semantics, sequential security typing, and sequential security-type preserving compilation that allow us enforcing security for multithreaded programs. Thanks to the modularity, there are no more restrictions on multithreaded source programs than on sequential ones, and yet we guarantee that their compilations are provably secure for a wide class of schedulers.
This work has been published in [12].

6.1.2. Secure Information Flow by Self-Composition

Information flow policies are confidentiality policies that control information leakage through program execution. A common means to enforce secure information flow is through information flow type systems. Although type systems are compositional and usually enjoy decidable type checking or inference, their extensibility is very poor: type systems need to be redefined and proven sound for each new single variation of security policy and programming language for which secure information flow verification is desired. In contrast, program logics offer a general mechanism to enforce a variety of safety policies, and for this reason are favored in Proof Carrying Code, a promising security architecture for mobile code. However, the encoding of information flow policies in program logics is not straightforward, because they refer to a relation between two program executions. The purpose of this work is to investigate logical formulations of secure information flow based on the idea of self-composition that reduces the problem of secure information flow of a program P to a safety property for a program P' derived from P.
This work will appear in the special issue of MSCS of PLID [27].

6.1.3. Robustness Guarantees for Anonymity

Anonymous communication protocols must achieve two seemingly contradictory goals: *privacy* (informally, they must guarantee the anonymity of the parties that send/receive information), and *robustness* (informally, they must ensure that the messages are not tampered). However, the long line of research that defines and analyzes the security of such mechanisms focuses almost exclusively on the former property and ignores the latter.
In this work, we initiate a rigorous study of robustness properties for anonymity protocols. We identify and formally define, using the style of modern cryptography, two related but distinct flavors of robustness. Our definitions are general (*e.g.*, they strictly generalize the few existent notions for particular protocols) and flexible (*e.g.*, they can be easily adapted to purely combinatorial/probabilistic mechanisms).
We demonstrate the use of our definitions through the analysis of several anonymity mechanisms (Crowds, broadcast-based mix-nets, DC-nets, Tor). Notably, we analyze the robustness of a protocol by Golle and Juels for the dining cryptographers problem, identify a robustness-related weakness of the protocol, and propose and analyze a stronger version.
This work has been published in [11].

6.1.4. Automated Code Injection Prevention for Web Applications

We propose a new technique based on multi-tier compilation for preventing code injection in web applications [29]. It consists in adding an extra stage to the client code generator which compares the dynamically generated code with the specification obtained from the syntax of the source program. No intervention from the programmer is needed. No plugin or modification of the web browser is required. The soundness and validity of the approach are proved formally by showing that the client compiler can be fully abstract. The practical interest of the approach is proved by showing that the actual implementation in the Hop environment imposes a performance penalty smaller than those of other techniques for preventing code injection.

6.1.5. Secure session calculi

We have proposed a secure information flow analysis for a session calculus based on the pi-calculus. More precisely, we have considered a calculus for multiparty sessions with delegation, enriched with security levels for both session participants and data. The calculus has also been equipped with a form of declassification for data, as required by most practical applications.

We have defined a type system for this calculus that ensures, besides the expected properties of session calculi (absence of communication errors and compliance to the protocol), also access control and secure information flow. Access control means that session participants cannot receive data of security level higher than or incomparable to their own. For instance, in a session involving a Customer, a Seller and a Bank, the secret credit card number of the Customer should be communicated to the Bank but not to the Seller. Secure information flow means that the confidentiality of data must be preserved throughout the session. In particular, our type system prevents any information leak through the specific constructs of the calculus, which are session opening, selection, branching and delegation. Finally, we showed that our type system allows exactly those declassifications that are permitted by the access control policy.

Our work has revealed an interesting interplay between the constraints of security type systems and those used in session types to ensure classical properties such as communication safety and session fidelity.

This work has been published in [15], and its full version, with complete definitions and proofs, is presented in [25].

6.2. Models and semantics

Participants: Gérard Berry, Gérard Boudol, Gustavo Petri, Christian Queinnec, Manuel Serrano.

6.2.1. Semantics of concurrent programming

The semantical framework of write-buffering is based on the work presented in 2009 at the POPL conference, with minor modifications. The framework of speculative computations is based on the work[14] with some major modifications. In particular, we have included an optimization related to write-buffering commonly found in current architectures. Also, two programming languages were considered in this framework, a high-level programming language, supporting locks for critical sections, and a low-level language, which only supports barriers and a simple compare-and-swap instruction. The interest of having these two languages is to show that the framework is well suited to modelling both the semantics of a high-level programming language (like ML for example), and the Instruction Set Architecture (ISA) of a machine architecture, which usually does not provide locks. In the thesis, Gustavo shows that the speculative semantics presented allows for many behaviors commonly found in relaxed memory model specifications. *Robustness* properties are provided guaranteeing that the speculative execution of programs that meet these properties coincide with their sequentially consistent execution.

To prove the utility of these two semantical frameworks, and to present existing memory models in an operational way, Gustavo instantiates both frameworks to model the TSO, PSO and RMO memory models of Sparc. In fact, the write-buffering framework is not fit for modelling RMO, which motivates the need for the more general (but more complex as well) framework of speculations. Having instantiations of both PSO and TSO in both frameworks makes their formal comparison possible. The main result of this work is a – non-trivial – proof that these instantiations coincide. This work has not been published yet.

6.2.2. Formal semantics for Hop

Following the continuation-based denotational semantics for a sequential subset for HOP [9], we have conceived a second formal semantics shaped as a small-step operational semantics [13] that covers a wider part of the language than the denotational one. It supports reasoning about web applications written in the multi-tier language HOP. In particular, it covers both server side and client side computations, as well as their interactions, and includes creation of web services, distributed client-server communications, concurrent evaluation of service requests at server side, elaboration of HTML documents, DOM operations, evaluation of script nodes in HTML documents and actions from HTML pages at client side.

6.3. Functional and concurrent programming

Participants: Pejman Attar, Frédéric Boussinot, Marcos Dione, Thomas Gazagnaire, Cyprien Nicolas, Manuel Serrano.

6.3.1. Bigloo

Software distribution: Three new major releases have been made available in 2010: Bigloo-3.3, 3.4, and 3.5. These releases have introduced the novelties:

- **crytography API** which implements traditional cryptography algorithms (rsa, dsa, aes, etc) and the PGP layer.
- **Android port** enables Bigloo to compile applications for the Android Smartphones equipped with Arm processors. This port supports native multi-threading.
- **Enhanced class support in the interpreter** eliminates most of the restrictions that applied to the object layer when used within the interpreter. That is, from now on, the compiled and interpreted code support the same constructs for object oriented programming.
- **Compilation improvements** that produce more efficient code for checking array bounds and dynamic types.

6.3.2. FunLoft

We have developped the language FunLoft along several axes: separate compilation and extension to higher-order functions (these are not first class members, but some limitations on them have been suppressed; for example, functions can be parameters; they still cannot be stored in memory). The new version v0.4 has been released. A book [24] has been written, entitled “Safe Reactive Programming - The FunLoft Language” which describes FunLoft and several applications (cellular automata, prey-predator systems, colliding particles, dataflow programming) in detail. We are currently implementing a simulation of physics which reflects several aspects of quantum mechanics such as self-interference, state superposition, and particles entanglement.

6.3.3. DSL

We have designed a kernel of an orchestration language, called DSL, which is translated in FunLoft. This parallel and reactive language is safe and we are in the process of implementing it in several other formalisms (SugarCubes, ReactiveML, Hop). Communication through the network will be based on HOP. The objective is to get a multi-formalisms platform for the programming of diffuse systems in heterogeneous environments. We are designing an extension of DSL to be able to maximize the use of cores by programs, preserving safety. In this proposal, we consider agents which are autonomous parallel entities which possess a memory. It is statically proved that each agent’s memory is free from interferences with other agents. Agents are executed on sites which are composed of several synchronised schedulers, mapped on distinct cores. The number of cores used by a site is dynamically adapted to the load of agents executed on the site.

6.3.4. Types and effects

We have designed a type and effect system and a deadlock-free semantics for shared memory concurrency. We implement it in an ML-like programming language which will be released under the name *Tesard*. We are writing a compiler of it, as a fork of *Llama Light*. (*Llama* is itself a fork of *OCaml* which simplifies the language and the type system and makes the compiler more modular.) The project is in its early stage of development.

6.4. Web programming

Participants: Marcos Dione, Zhengqin Luo, Cyprien Nicolas, Tamara Rezk, Bernard Serpette, Manuel Serrano.

6.4.1. Hop

Software distribution: Three new major branches have been released this year: HOP 2.0.x, 2.1.x, and 2.2.x. Here is short overview of the changes applied to the system.

- **Android port** which has consisted in porting the HOP runtime environment to the Android platform and also in creating a dedicated library that allows HOP to interact with the phone operating system. As such, it is for instance possible, from a HOP application, to react when a SMS is sent or received, when the orientation of the phone changes, etc. This is still an ongoing project. The HOP Android library still needs many extensions. This port greatly expands the range of devices were HOP can be run because the Android platform is mostly run in the ever-growing markets of cellular phones, netbooks and tablets.
- **Bundle OSGi** which allows HOP to be shipped in a bundle and executed as a independant component inside a Java Virtual Machine.
- **Integrated module system** which allows HOP client-side and server-side modules to be totally mixed up. Up to the branch 1.11.x client-side modules and server-side modules were disjoint entities. With versions 2.x client-side modules may *import* server-side modules and vice-versa. This major extension is a breakthrough toward a unified programming model.
- **Integrated macro system** which allows HOP macros to be defined on both ends of the application. Combined with the new *integrated module system* this is the second element that improves the coherence of the multi-tier programming paradigm advocated by HOP.
- **Improved browsers portability**, we have improved the portability of the client-side code generated by HOP. In particular, the generated code now smoothly works with Firefox, WebKit (*i.e.*, Safari and Google Chrome), and Opera. We have significantly improved the portability for Internet Explorer 8, but we are waiting the official version 9 of that navigator to conclude that effort.
- **Security Manager** which allows HOP to automatically detect cross-site scripting. This is the direct implementation of the formal studies about *Code Injection Prevention via Multitiers Compilation* that has not yet been published.
- **HSS**, the CSS for HOP has been completed. It extends CSS with user defined variables, functions, and element types. It can also be used with the Hop web development kit in which case, working hand in hand with the Hop programming language, it can be used to implement *skinning* or *theming* of web applications. HSS has been described in a conference paper [20].
- **Remote APIs** which allow HOP applications to import remote libraries, that is libraries made available independently of the HOP distribution. These remote APIs make it easy to implement *mashups* with HOP. Currently, we are proposing only a few number of these APIs but they cover a wide spectrum ranging from MAP API to QR code API, via multimedia APIs. All these libraries are available on the official HOP distribution site.

These elements are just some highlights amongst all the modifications and improvements applied to the system. A full list of changes can be found by inspecting the whole ChangeLog file included in the HOP package.
We also worked, within the Smartimmo DGE project, towards making Hop a good choice to develop applications that can inform their users about their whereabouts in a 'Smart Building' and allow them to control the environment. For this we first developed libraries to communicate with the WebServices offering such functionality, mainly through the SOAP, oBIX and STOMP protocols, widely used in the industry for such purposes.

Then we developed a demo that, thanks to the new Android port and the previously existing Maemo port, could run on smartphones. This demo not only made use of WebServices, but also accesed the sensors in the smartphone itself and used a Phidgets board with sensors attached to simulate a lighting level and temperature controlling application. For the latter we also developed a wrapper to the C library for accesing and controlling Phidget boards.

6.4.2. HopTeX

HopTeX is a new application for authoring HTML and LaTeX documents [30]. The content of the document is expressed in a blending of HTML and a dedicated wiki syntax. The rendering of the document is expressed by a set of CSS rules. The main originality of HopTex is to consider LaTeX as a new media type for HTML and to express the compilation from HTML to LaTeX by the means of dedicated style sheet rules.
This implementation extensively relies on two facilities generally only available on the client-side that HOP also supports on the server-side of the application: DOM manipulations and CSS server-side resolutions.

6.4.3. Mirage

Mirage [17] (http://www.openmirage.org/) is an open-source operating system for constructing secure, high-performance, reliable network applications across a variety of cloud computing and mobile platforms. Code can be developed on a normal OS such as Linux or MacOS X, and then compiled into a fully-standalone, specialized OS kernel that runs under the Xen hypervisor. Since Xen powers most public cloud computing infrastructure such as Amazon EC2 , this lets your servers elastically scale up massively with little effort on your part.)
Mirage is based around the OCaml language, with syntax extensions and libraries which provide networking, storage and concurrency support that are easy to use during development, and map directly into operating system constructs when being compiled for production deployment.
We have been working on using type-based meta-compilation techniques [[16],[8]] and embedded DSL (for html https://github.com/samoht/htcaml css https://github.com/samoht/cass as examples) to provide a platform for developping web-applications on top of mirage.

6.4.4. Orchestration

As more and more data becomes available over the web (through open databases, or web sites APIs for instance), new “mash-ups” applications appear daily. These applications aggregate the data available from remote web services in order to give a new view or a new usage of this data.
Such web applications are, de facto, distributed, and have to deal with all the typical issues of distributed programming, e.g. communications, synchronization, fault-tolerance, priorities. Another thing to note is that in a traditional grid computing environment, one developer has the control on all the resources used by his program, whereas it is generally not true inside the web environment.
These applications deal with more and more data sources, in case one would be unavailable, or outdated, so developers have to manage complex external interactions. Metaphorically speaking, they act as a web service conductors, that is why such new techniques are called “orchestration”.
Dedicated languages, so-called orchestration languages, provide new primitives to express such computations simply, and provide means to handle failures, timeouts and priorities. The orchestration language considered so far is Orc, developed at University of Texas at Austin.
We think that Hop can benefit from these primitives, offering a new, and we hope simple, way to program orchestration and mash-ups. Additionally, existing orchestration languages only provide those primitives on the server-side, Hop will permit to express orchestration in both server and client sides.
This is an on going thesis project, which is not yet part of the Hop standard distribution. We started by giving a scheme-oriented syntax of the Orc operators, and then rewrite the Orc semantics with this new syntax. Then the first implementation consisted of a direct translation of the semantics.
This implementation was difficult to achieve as Orc is actually a data-flow language, whereas Hop is not. We needed a model to represent how the data would flow from one instruction to another, using dedicated constructs.

We tried to model the semantics using Kahn networks. With this model, each Orc operator is a Kahn process, and those process are linked using streams.
This implementation is mainly composed of macros, it can run on top of the current Hop language, without having to modify Hop's runtime, or the Bigloo compiler itself to support those new constructs.

6.4.5. Ordered networks

Channels of communication in our ordered networds was oriented: in a channel from A to B, A can only write and B can only read. But, from the implementation point of view, sockets are used to design these channels. Since sockets are non oriented, we have studied how ordered networks can take benefits of this extra channel of communication. Behind the fact that the observed gain is not effective, the protocol for closing channels becomes not evident.
Ordered networks use an order to drive the routing process, but also a strategy for choosing the *best* neighboors. Generally, this strategy is based on the order and so on node's identifier of the network. We have studied a strategy based on the number of message sent on each channels a node have. When all nodes try to balance these numbers of message we observe good performances of the network, but without reaching the optimality. This strategy is particularly important since the performance of the routing process is no more related to a uniform distribution of the identifiers in the network.

7. Contracts and Grants with Industry

7.1. Contracts and Grants with Industry

7.1.1. DGE SmartImmo

The SmartImmo DGE project of the world class ICT cluster SCS (Solutions Communicantes Sécurisées) started in 2009. It aimed at controlling and reducing the costs associated with building management. The duration of the project is 2 years, and the funding is 75k Euros. In the context of this project, the INDES team will focus on making the Hop Web development kit compatible with industrial house automation systems.

7.1.2. All In My Music

Although this activity has not lead to a contract yet, we have collaborated with a local start-up named *All In My Music* that distributes music on-line. Using HOP we have been able to produce a portable software library that allows the company to ship music more easily on a wide set of devices.

8. Other Grants and Activities

8.1. National initiatives

8.1.1. STIC-AmSud FMCRYPTO

The FMCRYPTO project (for "Formal Methods for Cryptographically Secure Distributed Computations") is funded by the STIC-AmSud programme for 2 years, ending December 2010. The partners of this project are the INDES team at INRIA (coordinator: Tamara Rezk), Universidade Estadual de Campinas in Brazil, Universidad de Chile in Chile, and Universidad de la Republica in Uruguay.

8.1.2. COLOR project MATYSS

The project MATYSS (Models And TYpes for Secure Sessions), funded by the INRIA Sophia Antipolis COLOR programme, was extended to year 2010 (due to a late start in 2009). The partners of this project were the team INDES (coordinator) and the "Semantics and Logic of Computation" group at the Computer Science Department of the University of Torino.

8.1.3. ANR SETIN ParSec

The PARSEC project (for “Parallélisme et Sécurité”) has been funded by the ANR Sécurité Informatique programme for 4 years, starting January 2007. The partners of this project are the teams INDES formerly MIMOSA (coordinator) and EVEREST at INRIA Sophia Antipolis, LANDE at IRISA, MOSCOVA at INRIA Rocquencourt and CONCURRENCY at the PPS Laboratory of Paris 7 University and CNRS.

8.1.4. ANR DEFIS ParTout

The PARTOUT project (PARTOUT = PARallélisme parTOUT) is funded by the ANR DEFIS programme for 4 years, starting January 2009. The partners of this project are the teams INDES (coordinator), CNAM/Cédric, and LRI, Université d’Orsay.

8.1.5. ANR DEFIS PWD

The PWD project (for "Programmation du Web diffus") has been funded by the ANR Défis programme for 4 years, starting November 2009. The partners of this project are the teams INDES (coordinator), LIP6 at University Pierre et Marie Curie and PPS at University Denis Diderot.

9. Dissemination

9.1. Seminars and conferences

Gérard Berry held the Informatics and Digital Sciences Chair at Collège de France, which is co-financed by INRIA. His course was called “Penser, modéliser et maîtriser le calcul informatique”, in English “Thinking about, Modeling and Mastering Automatic Computation”. It consisted in a formal inaugural lecture, followed by eight lectures accompanied by eight seminars from external personalities, including a presentation of Diffuse Computing and HOP by Manuel Serrano. The inaugural lecture has been published as a book published in 2009 the lecture notes are available in [23]. The lecture and seminars are available in video on the Web, see [21]. Gérard Berry also repeated seven lectures in English at the University of Edinburgh and at the Royal Society of Edinburgh, accompanied by six seminars of French or British personalities, see [22]. The videos will be made available on the Web by INRIA. He also repeated five Collège de France lectures at ESIB Beyrouth. Gérard Berry gave 20 general conferences about the impact of Informatics in the Academic system, in Industry, in Science, and in the Education community, to academic or industrial audiences. He contributed to the introduction of Computer Science Teaching in the French education system by co-defining the Professor Training Program; he gave six conferences to French High Schools students and professors, and published a paper in the “Textes et documents pour la classe” education journal. [10].

Gérard Boudol participated in the meetings of the PARSEC project (in January, June and December), where he gave a talk on his current work with Gustavo Petri. He participated in the ETAPS Conferences, where the work [14] was presented. He participated in the ANR Colloque “Sytèmes embarqués, Sûreté de fonctionnement, Sécurité” in Toulouse where he presented the PARSEC project. He was a member of the jury of the PhD of Christelle Braun (INRIA COMETE), and of Michael Lienhardt (INRIA SARDES).

Ilaria Castellani visited the university of Torino for one week in January 2010, to pursue collaborative work within the MATYSS project. In September 2010, she attended the conference CONCUR 2010 in Paris, where she presented the MATYSS work [15]. She also presented this work at one of the meetings of the ANR project PARSEC.

Thomas Gazagnaire has been invited to give a talk on his experience on using functional programing in an industrial context at the Ocaml User Meeting in April 2010 and in the INRIA Sardes team meeting in June 2010. An experience report [19] on this topic has been accepted in the 15th ACM SIGPLAN International Conference on Functional Programming (ICFP 2010). A general overview of Mirage [17] has been published in the 2nd USENIX Worshop on Hot Topics in Cloud Computing in June 2010. The work about type-based meta-compilation [16] has been accepted in March 2010 at the 1st Workshop on Generative Technologies (WGT'2010) and will be published [8] in the Electronic Notes in Theoretical Computer Science. He Participated to the meta-programing BOF in the Commercial User of Functional Programming (CUFP 2010) conference. Also, some details about Mirage's ML micro-kernels have been presented to the 2010 Workshop on ML in September 2010.

Johan Grande gave a presentation of region-based memory management at the Journées FedeRez 2010.

Zhengqin Luo participated the FMcrypto workshop in Chile in April 2010, where he presented the work of robustness guarantee of anonymity. In April 2010, he attented the Cosyproof workshop held in Barbizon. In June 2010, he participated the PLDI conference and APLWACA workshop held in Toronto, and gave a talk on the work of operational semantics of Hop. In July 2010, he participated the FLOC conference held in Edinburgh, and he also presented the work of anonymity at CSF.

Gustavo Petri attended the ESOP Conference in Cyprus and presented the work [14].

Christian Queinnec gave an invited presentation at the JFLA'10 conference. Christian Queinnec has been working on an architecture for mechanized grading: a series of protocols rules a constellation of servers to offer robustness and tolerance to malicious students' programs. This work was presented at CEDU2010 [18].

Tamara Rezk attended a PARSEC meeting in January 2010, where she presented her work on security for Hop applications. She attended the FMcrypto workshop in Chile in April 2010, where she presented her work on code injection for web applications. In April 2010, she participated the Cosyproof workshop held in Barbizon. In November, she presented her work on security for web applications at IST Lisbon. In December, she participated to the final PARSEC and FMCrypto meetings in Paris and Campinas respectively.

Manuel Serrano gave several presentations of the HOP system. First, he gave an invited talk at the European Lisp workshop, a join event with Ecoop 2010. Then, he gave a seminar about HOP at Microsoft Research Cambridge in June. Finally, he presented diffuse programming in one of the *Seven keys to the digital future*, conferences organized by the Royal Society of Edinburgh in Scotland.

9.2. Animation

Gérard Boudol was a reviewer for the PhD thesis of Jade Alglave (INRIA MOSCOVA). He was also a member of the jury for a competition for a “Maître de Conférence” position at Paris 7 University.

Frédéric Boussinot is the coordinator of the ANR DEFIS project PARTOUT. He was a referee and member of the Jury pf the PhD Thesis of Luidnel Maignan (University Paris-Sud 11).

Ilaria Castellani was the coordinator of the COLOR project MATYSS.

Tamara Rezk is the coordinator of the STIC-AmSud FMCrypto project. She was a member of the programme committees VS-Theory, SAFA (co-chair), and ACM SAC SV track. She is a member of the Committee of Courses and Coloquiums at Inria. She participates as a selecteur for the platform MyLibrary at Inria. She was a reviewer for ICFP'10, CCS'10, Esorics'10, and the Journal of Logical Methods in Computer Science.

Manuel Serrano is the coordinator of the ANR DEFIS project PWD. He serves on the program committees of the following conferences:

ICDCIT 2011 7th International Conference on Distributed Computing and Internet Technology.

ILC 2010 International Lisp Conference.

TOOLS 2010 48th International Conference on Objects, Models, Components, Patterns.

ICFP 2010 15th ACM SIGPLAN International Conference on Functional Programming.

ECOOP 2010 24th European Conference on Object-Oriented Programming.

JFLA 2010 21th journées francophones des langages applicatifs.

Manuel Serrano was a member of the jury of the PhD Thesis of Charlotte Herzeel (Brussels) and he was a referee and member of the jury of the PhD thesis of Floréal Morandat (Montpellier).

9.3. Teaching

Frédéric Boussinot supervised the summer internship of Pejman Attar, a graduate student of the University of Paris 7, Master MPRI, who worked during 6 months on the language DSL.

Johan Grande is giving 1st and 2nd year algorithmics and programming (Python and Java) tutorial/practical classes at Polytech'Nice (76 hours altogether).

Cyprien Nicolas supervised 30 hours of practical exercises in the course “Introduction to Functionnal Programming” at the University of Nice.

Christian Queinnec is the directory of the the École doctorale EDITE of the University *Pierre et Marie Curie* and he was appointed as action lead on european doctoral school within the EIT KIC ICT. This year, Christian Queinnec delivered two lectures: one in Master1 on compilation where a programming language similar to Javascript (with XML as concrete syntax) is interpreted and compiled into C (both the interpreter and compiler are written in Java). The other lecture in Master2 is on the architecture of web applications.

Tamara Rezk is teaching a course on provable cryptography in the master “Cryptographie et Securit'e” at University of Nice-Sophia Antipolis. She supevised a project of the student Inza Bamba, from the master of “Ubiquitous Networking and Computing” and a Phd student, Zhengqin Luo. In November, she taught a master course on Proof Carrying Code in IST Lisbon.

Manuel Serrano gave a seminar about diffuse and HOP programming in the seminar series associated with the course of Gérard Berry at Collège de France called “Penser, modéliser et maîtriser le calcul informatique”. He supervised the summer internship of Cyprien Nicolas, a graduate student of the University of Nice, Master Ubinet, who worked during 6 months on preparing the integration of the orchestration language Orc and the multi-tier programming language Hop. Manuel Serrano, in collaboration with Christian Queinnec, published an article in the French magazine *Linux Magazine France*.

10. Bibliography

Major publications by the team in recent years

[1] G. BARTHE, T. REZK, A. RUSSO, A. SABELFELD. *Security of Multithreaded Programs by Compilation*, in "ESORICS", 2007, p. 2-18.

[2] G. BOUDOL, I. CASTELLANI. *Noninterference for Concurrent Programs and Thread Systems*, in "Theoretical Computer Science", 2002, vol. 281, n^{o} 1, p. 109-130.

[3] C. FOURNET, T. REZK. *Cryptographically sound implementations for typed information-flow security*, in "Proceedings of the 35th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2008", San Francisco, California, USA, January 7-12 2008, p. 323-335.

[4] M. SERRANO, E. GALLESIO, F. LOITSCH. *HOP, a language for programming the Web 2.0*, in "Proceedings of the First Dynamic Languages Symposium", Portland, Oregon, USA, October 2006.

[5] M. SERRANO. *Bee: an Integrated Development Environment for the Scheme Programming Language*, in "Journal of Functional Programming", May 2000, vol. 10, n^{o} 2, p. 1–43.

Publications of the year

Doctoral Dissertations and Habilitation Theses

[6] G. PETRI. *Operational Semantics of Relaxed Memory Models*, Université de Nice - Sophia Antipolis, November 2010.

Articles in International Peer-Reviewed Journal

[7] G. BOUDOL. *Typing termination in a higher-order concurrent imperative language*, in "Information and Computation", 2010, vol. 208, p. 716-736.

[8] T. GAZAGNAIRE, A. MADHAVAPEDDY. *Dynamics for ML using Meta-Programming*, in "Electonic Notes in Theoritical Computer Science", 2010.

[9] M. SERRANO, C. QUEINNEC. *A multi-tier semantics for Hop*, in "Higher Order and Symbolic Computation (HOSC)", 2010 [*DOI :* 10.1007/s10990-010-9061-9].

Articles in National Peer-Reviewed Journal

[10] G. BERRY. *Seven Keys to the Digital Future*, in "Textes et documents pour la classe", 2010, n^{o} 997.

International Peer-Reviewed Conference/Proceedings

[11] G. BARTHE, A. HEVIA, Z. LUO, T. REZK, W. B.. *Robustness Guarantees for Anonymity*, in "CSF", 2010, p. 91-106.

[12] G. BARTHE, T. REZK, A. RUSSO, A. SABELFELD. *Security of Multithreaded Programs by Compilation*, in "ESORICS'07 Special Issue in ACM Transactions on Information and System Security (TISSEC)", 2010.

[13] G. BOUDOL, Z. LUO, T. REZK, M. SERRANO. *Towards Reasoning for Web Applications: an Operational Semantics for Hop*, in "Proceedings of the first Workshop on Analysis and Programming Languages for Web Applications and Cloud Applications (APLWACA'10)", Toronto, Canada, Jun 2010.

[14] G. BOUDOL, G. PETRI. *A Theory of Speculative Computation*, in "ESOP", LNCS, 2010, p. 165-184.

[15] S. CAPECCHI, I. CASTELLANI, M. DEZANI-CIANCAGLINI, T. REZK. *Session Types for Access and Information Flow Control*, in "CONCUR'10", P. GASTIN, F. LAROUSSINIE (editors), Lecture Notes in Computer Science, Springer Verlag, 2010, vol. 6269, p. 237-252.

[16] T. GAZAGNAIRE, A. MADHAVAPEDDY. *Statically-typed value persistence for ML*, in "Proceedings of the Workshop on Generative Technologies (WGT'2010)", March 2010.

[17] A. MADHAVAPEDDY, R. MORTIER, R. SOHAN, T. GAZAGNAIRE, S. HAND, T. DEEGAN, D. MCAULEY, J. CROWCROFT. *Turning down the LAMP: software specialisation for the cloud*, in "Proceedings of the 2nd USENIX conference on Hot topics in cloud computing", Berkeley, CA, USA, HotCloud'10, USENIX Association, 2010, p. 11–11.

[18] C. QUEINNEC. *An Infrastructure for Mechanised Grading*, in "CSEDU 2010 – Proceedings of the second International Conference on Computer Supported Education", Valencia, Spain, Apr 2010, vol. 2, p. 37–45, Cf. archive HAL.

[19] D. SCOTT, R. SHARP, T. GAZAGNAIRE, A. MADHAVAPEDDY. *Using functional programming within an industrial product group: perspectives and perceptions*, in "Proceedings of the 15th ACM SIGPLAN international conference on Functional programming", New York, NY, USA, ICFP '10, ACM, 2010, p. 87–92 [*DOI :* 10.1145/1863543.1863557].

[20] M. SERRANO. *HSS: a Compiler for Cascading Style Sheets*, in "12th Sigplan International Conference on Principles and Practice of Declarative Programming (PPDP)", Hagenberg, Austria, Jul 2010.

Scientific Books (or Scientific Book chapters)

[21] G. BERRY. *Penser, modéliser et maîtriser le calcul informatique, vidéos*, 2010, http://www.college-de-france.fr/default/EN/all/cha_inf2009.

[22] G. BERRY. *Seven Keys to the Digital Future*, 2010, http://idea.ed.ac.uk/future/.

[23] G. BERRY. *Penser, modéliser et maîtriser le calcul informatique, notes de cours*, in "Annales du Collège de France", Collège de France, 2010.

[24] F. BOUSSINOT. *Safe Reactive Programming - The FunLoft Language*, Lambert Academic Pub., 2010.

Research Reports

[25] S. CAPECCHI, I. CASTELLANI, M. DEZANI-CIANCAGLINI, T. REZK. *Session Types for Access and Information Flow Control*, Inria, 2010, http://hal.archives-ouvertes.fr/inria-00511304/PDF/RR-7368.pdf.

Scientific Popularization

[26] M. SERRANO, C. QUEINNEC. *HTML5 Video portable avec Heb*, in "Gnu Linux Magazine France", Jan 2010, n^{o} 129.

Other Publications

[27] G. BARTHE, P. D'ARGENIO, T. REZK. *Secure Information Flow by Self Composition*, 2010, to appear.

[28] F. BOUSSINOT. *The FunLoft Language*, 2010.

[29] Z. LUO, T. REZK, M. SERRANO. *Automated Code Injection Prevention for Web Applications*, 2010, Submitted.

[30] M. SERRANO. *HopTeX - Compiling HTML to LaTeX with CSS*, 2010, Submitted.