Activity Report 2015

Project-Team INDES

Secure Diffuse Programming

RESEARCH CENTER
Sophia Antipolis - Méditerranée

THEME
Distributed programming and Software engineering

Table of contents

Project-Team INDES

Creation of the Team: 2009 January 01, updated into Project-Team: 2010 July 01

Keywords:

Computer Science and Digital Science:

1.3. - Distributed Systems
2. - Software
2.1. - Programming Languages
2.1.3. - Functional programming
2.1.7. - Distributed programming
2.1.8. - Synchronous languages
2.1.9. - Dynamic languages
2.2.1. - Static analysis
2.2.3. - Run-time systems
4. - Security and privacy
4.3.3. - Cryptographic protocols
4.6. - Authentication
4.7. - Access control
4.8. - Privacy-enhancing technologies

Other Research Topics and Application Domains:

6.3.1. - Web
6.4. - Internet of things
9.4.1. - Computer science
9.8. - Privacy

1. Members

Research Scientists

Manuel Serrano [Team leader, Inria, Senior Researcher, HdR]
Nataliia Bielova [Inria, Researcher]
Ilaria Castellani [Inria, Researcher]
Tamara Rezk [Inria, Researcher]
Bernard Serpette [Inria, Researcher]

Engineers

Cédric Duminy [Inria, from May 2015]
Vincent Prunet [Inria]
Erwan Demairy [Inria, until Oct 2015]

PhD Students

Yoann Couillec [Inria, until Oct 2015]
Francis Doliére Some [Inria, from Nov 2015]
Colin Vidal [Inria, from Jul 2015]

Visiting Scientist

Vineet Rajani [MPI, until Mar 2015]

Administrative Assistant

Nathalie Bellesso [Inria]

Others

Gérard Boudol [Emeritus Researcher]
Francis Doliére Some [Inria, intern, from Jul 2015 until Sep 2015]
Rohan Katyal [Inria, intern, from Jun 2015 until Aug 2015]
Diana Ioana Proteasa Nicola [M2 intern, until Jan 2015]

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. Research Program

3.1. Parallelism, concurrency, and distribution

Concurrency management is at the heart of diffuse programming. Since the execution platforms are highly heterogeneous, many different concurrency principles and models may be 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 [6]. We have created a 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 [7]. 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 former Inria teams [2], [1]. 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 [4]. 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 for daily 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 that 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. Robotics

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 robot as a web entity, and the use of remote web services to manage, monitor or extend the features of the robot. Among the direct benefits of relying on a web framework for robotics are the ability to use any web enabled device such as a smartphone or tablet to drive the robot.

5. New Software and Platforms

5.1. Web programming

Participants: Yoann Couillec, Colin Vidal, Vincent Prunet, Manuel Serrano [correspondant].

5.1.1. The HOP web programming environment

HOP is a 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. It is released under the GPL license. It is available at http://hop.inria.fr.

- Participants: Manuel Serrano
- Contact: Manuel Serrano
- URL: http://hop.inria.fr

5.1.2. The Bigloo compiler

The programming environment for the Bigloo compiler [7] 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.

- Participants: Manuel Serrano
- Contact: Manuel Serrano
- URL: http://www-sop.inria.fr/teams/indes/fp/Bigloo

6. New Results

6.1. Web programming

Participants: Yoann Couillec, Vincent Prunet, Manuel Serrano [correspondant].

6.1.1. Hop.js

Multitier programming languages unify within a single formalism and a single execution environment the programming of the different tiers of distributed applications. On the Web, this programming paradigm unifies the client tier, the server tier, and, when one is used, the database tier. This homogenization offers several advantages over traditional Web programming that rely on different languages and different environments for the two or three tiers of the Web application: programmers have only one language to learn, maintenance and evolution are simplified by the use of a single formalism, global static analyses are doable as a single semantics is involved, debugging and other runtime tools are more powerful as they access global informations about the execution.

The three first multitier platforms for the Web all appeared in 2006: GWT (a.k.a., Google Web Toolkit), Links, and Hop [6], [5]. Each relied on a different programming model and languages. GWT maps the Java programming model on the Web, as it allows, Java/Swing likes programs to be compiled and executed on the Web; Links is functional language with experimental features such as the storing of the whole execution context on the client; Hop is based on the Scheme programming language. These three pioneers have open the path for the other multitier languages such as, Ocsigen for Ocaml, UrWeb, js-scala, etc.

In spite of their interesting properties, multitier languages have not become that popular on the Web. Today, only GWT is widely used in industrial applications but arguably GWT is not a fully multitier language as developing applications with GWT requires explicit JavaScript and HTML programming. This lack of popularity of other systems is likely due to their core based languages than to the programming model itself.

JavaScript is the *defacto* standard on the Web. Since the mid 90's, it is the language of the client-side programming and more recently, with systems like Node.js, it is also a viable solution for the server-side programming. As we are convinced by the virtues of multitier programming we have started a new project consisting of enabling multitier programming JavaScript. We have created a new language called HopScript, which is a minimalist extension of JavaScript for multitier programming, and we have implemented a brand new runtime environment called Hop.js. This environment contains a builtin Web server, on-the-fly HopScript compilers, and many runtime libraries.

HopScript is a super set of JavaScript, *i.e.,* all JavaScript programs are legal HopScript programs. Hop.js is a compliant JavaScript execution environment as it succeeds at 99% of the Ecma 262 tests suite. The Hop.js environment also aims at Node.js compatibility. In its current version it supports about 70% of the Node.js runtime environment. In particular, it fully supports the Node.js modules, which lets Hop programs reuse existing Node.js modules as is.

After a full year of active development to enhance JavaScript and Node.js compatibility, to incorporate features of JavaScript 1.6, and to design new language constructs for machine-to-machine communication, we are now ready to release Hop.js. This will appear at the beginning of 2016.

6.1.2. Data source

During the past few years the volume of accumulated data has increased dramatically. New kinds of data stores have emerged as NoSQL family stores. Many modern applications now collect, analyze, and produce data from several heterogeneous sources. However implementing such applications is still difficult because of lack of appropriate tools and formalisms. We propose a solution to this problem in the context of the JavaScript programming language by extending array comprehensions. Our extension allows programmers to query data from usual stores, such as SQL databases, NoSQL databases, Semantic Web data repositories, Web pages, or even custom user defined data structures. The extension has been implemented in the Hop.js system. It has been described in the paper [10], which has been presented at the ACM DBPL'15 conference.

6.2. Distributed programming

Participants: Gérard Boudol, Johan Grande, Manuel Serrano [correspondant].

Shared-memory concurrency is a classic concurrency model which, among other things, makes it possible to take advantage of multicore processors that are now widespread in personal computers. Concurrent programs are prone to deadlocks which are notoriously hard to predict and debug. Programs using mutexes, a very popular synchronization mechanism, are no exception.

We have studied deadlock avoidance methods with the aim of making programming with mutexes easier. We first studied a method that uses a static analysis by means of a type and effect system, then a variation on this method in a dynamically typed language.

We developed more the second method. It mixes deadlock prevention and avoidance to provide an easy-to-use and expressive deadlock-free locking function. We implemented it as a Hop library. This lead us to develop a starvation-free algorithm to simultaneously acquire an arbitrary number of mutexes, and to identify the concept of asymptotic deadlock. While doing so, we also developed an optimization of exceptions (finally blocks).

Our performance tests seem to show that using our library has negligible impact on the performance of real-life applications. Most of our work could be applied to other structured programming languages such as Java.

This work has been presented at the 17th International Symposium on Principles and Practice of Declarative Programming (PPDP'15) [13]. More details can be found in Grande's PhD thesis [8].

6.3. Types

Participants: Ilaria Castellani, Bernard Serpette.

6.3.1. Behavioural Types

The survey paper https://hal.inria.fr/hal-01213201 presents a state-of-the-art of a recent trend of research on the use of behavioural types for specifying and analysing security properties of communication-centred systems. It is essentially an outcome of the working group on security of the BETTY COST Action, and it offers a unified overview of various proposals that have been put forward in the last few years, both within the BETTY community and outside it, to combine security analysis with behavioural types.

6.3.2. Abstract Rewriting Systems

We have formalised, with the Coq system, the beginning of Paul-André Melliès's thesis concerning abstract rewriting systems. Behind the interest of studying rewriting systems, which are the roots of all small step semantics of programming languages, this particular formalisation was attractive since it gives a concrete example where we have to manage dependant types.

This was done in collaboration with Eduardo Bonelli and Pablo Barenbaum of University of Quilmes, Argentina. The specification and the proofs of this work take 2200 lines of Coq.

6.4. Security

Participants: Ilaria Castellani, Francis Doliere Some, Nataliia Bielova, Bernard Serpette, Tamara Rezk [correspondant].

6.4.1. Hybrid Typing of Secure Information Flow in a JavaScript-like Language

We propose a novel type system for securing information flow in a core of JavaScript. This core takes into account the defining features of the language, such as prototypical inheritance, extensible objects, and constructs that check the existence of object properties. We design a hybrid version of the proposed type system. This version infers a set of assertions under which a program can be securely accepted and instruments it so as to dynamically check whether these assertions hold. By deferring rejection to runtime, the hybrid version can typecheck secure programs that purely static type systems cannot accept.

This work has been published at the 10th International Symposium on Trustworthy Global Computing [11].

6.4.2. Modular Monitor Extensions for Information Flow Security in JavaScript

Client-side JavaScript programs often interact with the web page into which they are included, as well as with the browser itself, through APIs such as the DOM API, the XMLHttpRequest API, and the W3C Geolocation API. Precise reasoning about JavaScript security must therefore take API invocation into account. However, the continuous emergence of new APIs, and the heterogeneity of their forms and features, renders API behavior a moving target that is particularly hard to capture. To tackle this problem, we propose a methodology for modularly extending sound JavaScript information flow monitors with a generic API. Hence, to verify whether an extended monitor complies with the proposed noninterference property, our methodology requires only to prove that the API satisfies a predefined set of conditions. In order to illustrate the practicality of our methodology, we show how an information flow monitor-inlining compiler can take into account the invocation of arbitrary APIs, without changing the code or the proofs of the original compiler. We provide an implementation of such a compiler with an extension for handling a fragment of the DOM Core Level 1 API. Furthermore, our implementation supports the addition of monitor extensions for new APIs at runtime. This work has been published at the 10th International Symposium on Trustworthy Global Computing [12].

6.4.3. Relaxed Noninterference

We have began a study concerning the use of gradual typing for down casting or declassification for information flow. The particularity of this work is to use a finite state machine to gradually accept the down casting process.

This work is done with Éric Tanter of University of Santiago de Chile, in the context of the project Conicyt Redes CEV Challenges on Electronic Voting.

6.4.4. Hybrid Monitoring of Attacker knowledge

Enforcement of non-interference requires to prove that an attacker's knowledge about the initial state remains the same after observing a programs public output. We define a powerful hybrid monitoring mechanism which evaluates dynamically the knowledge that is contained in program variables. To get a precise estimate of the knowledge, the monitor statically analyses non-executed branches. We show that our knowledge-based approach can be combined with existing dynamic monitors for non-interference. A distinguishing feature of such a combination is that the combined monitor is provably more powerful than each mechanism taken separately. We demonstrate this by proposing a knowledge-enhanced version of a dynamic monitor based on the no-sensitive-upgrade principle. We show how to use the knowledge computed by our hybrid monitor to quantify information leakage associated to the program output. The monitor and its static analysis has been formalized and proved correct within the Coq proof assistant.

6.4.5. A Taxonomy of Information Flow Monitors

We propose a rigorous comparison of information flow monitors with respect to two dimensions: soundness and transparency.

For soundness, we notice that the standard information flow security definition called *Termination-Insensitive Non-interference (TINI)* allows the presence of termination channels, however it does not describe whether the termination channel was present in the original program, or it was added by a monitor. We propose a stronger notion of noninterference, that we call *Termination-Aware Non-interference (TANI)*, that captures this fact, and thus allows us to better evaluate the security guarantees of different monitors. We further investigate TANI, and state its formal relations to other soundness guarantees of information flow monitors. For transparency, we identify different notions from the literature that aim at comparing the behaviour of monitors. We notice that one common notion used in the literature is not adequate since it identifies as better a monitor that accepts insecure executions, and hence may augment the knowledge of the attacker. To discriminate between monitors' behaviours on secure and insecure executions, we factorized two notions that we call true and false transparency. These notions allow us to compare monitors that were deemed to be incomparable in the past.

We analyse five widely explored information flow monitors: no-sensitive- upgrade (NSU), permissive-upgrade (PU), hybrid monitoring (HM), se- cure multi-execution (SME), and multiple facets (MF).

This work has been accepted for publication in the International Conference on Principles of Security and Trust (POST 2016).

6.4.6. A Study of JavaScript constructs used in Top Alexa Sites

Several works on JavaScript analysis have shown that including remote scripts can introduce severe security implications in the behavior of the whole web application. To deal with different kinds of attacks, a number of research groups are developing automatic tools to analyze JavaScript programs. However, most of these works rely on one assumption: the scripts are written in a subset of JavaScript language meaning that only certain constructs are used (that are easier to analyse automatically) and others are omitted (for example, eval is impossible to analyze statically). The goal of the internship was to account for the use of each JavaScript construct in real world programs. To achieve that, we first did a large-scale crawl of the top 10,000 Alexa sites, collecting both inlined scripts and remote scripts. Second, we established the popularity of remote scripts. Next, we accounted for the occurrence of JavaScript constructs in the collected programs. Finally, we use the occurrence of different constructs as basis to propose a subset of JavaScript language, which covers most of JavaScript programs found in the wild. One can rely on this evidence-based subset of JavaScript in future works on that language.

7. Partnerships and Cooperations

7.1. National Initiatives

7.1.1. ANR AJACS

The AJACS project (Analyses of JavaScript Applications: Certification & Security) has been funded by the ANR for 42 months, starting December 2014. The goal of AJACS project is to provide strong security and privacy guarantees on the client side for web application scripts. The Indes members are involved in the tasks WP2 Certified Analyses and WP3 Security of JavaScript Applications. The partners of this project include Inria teams Celtique (coordinator), Toccata, and Prosecco.

7.1.2. FUI UCF

The 3 years long UCF project aims at developing a reactive Web platforms for delivering multimedia contents. The partners of the project are the startups Alterway, OCamlPro, and XWiki, and the academic research laboratories of University Pierre et Marie Curie and Denis Diderot.

7.2. European Initiatives

7.2.1. FP7

7.2.1.1. MEALS

Title: Mobility between Europe and Argentina applying Logics to Systems

Program: FP7

Instrument: International Research Staff Exchange Scheme

Duration: October 2011 - September 2015

Coordinator: Pedro D'Argenio

Partners:

Imperial College of Science, Technology and Medicine (United Kingdom)

Rheinisch-Westfaelische Technische Hochschule Aachen (Germany)

Technische Universiteit Eindhoven (Netherlands)

Technische Universitaet Dresden (Germany)

University of Leicester (United Kingdom)

Universitaet Desarlandes (Germany)

Universidad de Córdoba (Argentina)

Universidad de Buenos Aires (Argentina)

Inria contact: Castuscia Palamidessi

Abstract:The MEALS project (Mobility between Europe and Argentina applying Logics to Systems) goals cover three aspects of formal methods: specification (of both requirement properties and system behavior), verification, and synthesis. The Indes members are involved in the task of Security and Information Flow Properties (WP3). The partners in this task include University of Buenos Aires, University of Córdoba, Inria (together with Catuscia Palamidessi, Kostas Chatzikokolakis, Miguel Andrés) and University of Twente. The web page of the project can be found at http://www.meals-project.eu.

7.2.1.2. RAPP

Program: http://rapp-project.eu

Title: Robot App Store

Collaborator: Inria Coprin

Abstract: RAPP is a 36 months pan-european FP7 project, started in December 2013. Hop is used in the development of prototypes of the Coprin Ang rollator transfer device, for mobility assistance and activity monitoring.

7.2.2. Collaborations in European Programs, except FP7 & H2020

Program: ICT Cost Action IC1201

Project acronym: BETTY

Project title: Behavioural Types for Reliable Large-Scale Software Systems

Duration: October 2012 - October 2016

Coordinator: Simon Gay, University of Glasgow

Other partners: Several research groups, belonging to 22 european countries

Abstract: The aim of BETTY is to investigate and promote behavioural type theory as the basis for new foundations, programming languages, and software development methods for communication-intensive distributed systems. Behavioural type theory encompasses concepts such as interfaces, communication protocols, contracts, and choreography.

Program: ICT Cost Action IC1405

Project title: Reversible computation - extending horizons of computing

Duration: November 2014 - November 2018

Coordinator: Irek Ulidowski, University of Leicester

Abstract: Reversible computation is an emerging paradigm that extends the standard forwards mode of computation with the ability to execute in reverse. It aims to deliver novel computing devices and software, and to enhance traditional systems. The potential benefits include the design of reversible logic gates and circuits - leading to low-power computing and innovative hardware for green ICT, new conceptual frameworks and language abstractions, and software tools for reliable and recovery-oriented distributed systems.

This Action is the first European network of excellence aimed at coordinating research on reversible computation.

7.3. International Research Visitors

7.3.1. Visits of International Scientists

7.3.1.1. Internships

Vineet Rajani

Date: December 2014 - March 2015

MPI (Germany)

Katyal Rohan

Date: June 2015 - Aug 2015

Institution: IIIT-D (India)

Francis Dolière Some

Date: July 2015 - Sept 2015

University of Ouagadougou (Burkina)

8. Dissemination

8.1. Promoting Scientific Activities

8.1.1. Scientific events organisation

8.1.1.1. General chair, scientific chair

Manuel Serrano was the General chair of the 11th ACM Dynamic Language Symposium (DLS'15), co-located with Splash 2015 (http://2015.splashcon.org/track/dls2015). Manuel Serrano was the General chair of 13th Trends In Functional Programming Symposium. (https://tfp2015.inria.fr/).

8.1.1.2. Member of the organizing committees

Nataliia Bielova was a publication chair of International Symposium on Engineering Secure Software and Systems (ESSoS 2015). Ilaria Castellani co-organised the workshop TRENDS 2015, affiliated with the CONCUR 2015 conference which took place in Madrid at the beginning of September.

8.1.2. Scientific events selection

8.1.2.1. Member of the conference program committees

- Ilaria Castellani served on the programme committee member of the conferences CONCUR 2015, TGC 2015, TTCS 2015 and ICTAC 2015.
- Tamara Rezk served on the programme committee member of the conferences ESSOS 2015, CSF 2015, POST 2015, PLAS 2015.
- Manuel Serrano served on the program committee of the 20th ACM ICFP'15.

8.2. Teaching - Supervision - Juries

8.2.1. Teaching

Master : Nataliia Bielova, Information Flow Security in Web Applications, 15 ETD, University of Pierre et Marie Curie, France

Doctorat : Ilaria Castellani, Behavioral Types, 15H ETD, University of Florence, Italy

Licence : Vincent Prunet, Algorithms and Data Structures, 80 ETD, L2, Lycée International de Valbonne Sophia Antipolis (within the scope of the national Inria action to promote early CS courses in all scientific curricula), France

Master : Tamara Rezk, Web Application Security, 28H ETD, University of Nice Sophia Antipolis, France

Master : Tamara Rezk, Proofs of Cryptography, 28H ETD, University of Nice Sophia Antipolis, France

Master : Tamara Rezk, Information Flow Security in Web Applications, 15 ETD, University of Pierre et Marie Curie, France

Master: Bernard Serpette, From Lambda-calculus and Pi-calculus to an Abstract Distributed Machine, 24H ETD, Escuela de Verano de Ciencias Informàticas, Rio Cuarto, Argentina

Master: Manuel Serrano, Programming the Diffuse Web, 4,5h ETD, Ecole Normale Supérieure de Cachan, France

8.2.2. Supervision

PhD : Johan Grande, Conception and Implementation of a modular concurrent programming language, University of Nice Sophia Antipolis, September 2015, Manuel Serrano and Gérard Boudol

PhD in progress: Yoann Couillec, Langages de programmation et données ouvertes, University of Nice Sophia Antipolis, 1/10/2012, Manuel Serrano

PhD in progress : Francis Doliére Some, Web Tracking Prevention, November 2015, Nataliia Bielova and Tamara Rezk

PhD in progress : Colin Vidal, HipHop, September 2015, Manuel Serrano and Gérard Berry

PFE master 2: Julien Chiramello, Décomposition en nombres premiers en informatique quantique, Tamara Rezk and Benjamin Grégoire

8.2.3. Juries

Ilaria Castellani was a member of the "Comité de sélection" for a position of Maître de conférences at the University of Paris-Est Créteil (Paris 12).

Ilaria Castellani was an examiner of the PhD thesis of Ioana Cristescu, Université Paris-Diderot (Paris 7).

Tamara Rezk was an examiner of the PhD thesis of Marco Patrignani, KU Leuven University.

Manuel Serrano was a member of a "comité de sélection" of UPMC (Paris 6).

8.3. Transfer

8.3.1. Price discrimination in e-commerce

In 2014, we started acollaborative project with Thomas Vissers, Wouter Joosen (KULeuven, Belgium) and Nick Nikiforakis (Stony Brook University, USA). The goal of this project is to analyse the price discrimination in e-commerce, and more precisely in the online airline tickets websites. This work (published at HotPETs 2014) has achieved a strong impact on the general public and society at large – its scientific results has been disseminated via the popular science French magazine "Science et vie" No. 1177, in the article called "*Achat sur Internet – Des prix á la tête du client!*".

8.3.2. WebRobotics

The WebRobotics initiative aims at developing collaborations with partner academic and industry teams to jointly prototype and experiment end user applications involving assistive robots and sensor devices (depending on the size and number of the embedded components, applications may be either classified as robotic or IoT ones). Each WebRobotics project is structured around partner medical institutions that provide key requirements to specifications and use the actual prototype throughout their daily activity. WebRobotics Applications all use Hop.js as their core framework, natively supporting web protocols for communication and distribution of tasks, and any web enabled device such as a smartphone or tablet to drive the robots and applications. In 2015, The initiative accounted to two full time engineers.

The Top Three Benefits of WebRobotics:

- WebRobotics focuses on key societal issues, developing real applications for demanding users.
- Application developers and users feedback to Hop.js framework developers, helping identify and prioritize key requirements.
- The WebRobotics application portfolio fosters the dissemination and transfer of the Hop.js technology to the Industry.

The WebRobotics initiative now encompasses several prototypes in use by medical foundations and hospitals.

- RAPP. The WebRobotics project is now part of the RAPP FP7 european project, launched in December 2013, where Hop.js technology is used by several academic and SME R&D teams to develop a distributed software platform and applications for assistive robotics. Two prototypes are being developed, the first one is a personal coach robot (a Nao humanoid robot embedding Hop.js distributed applications), and the second one is a smart rollator (a walking aid with additional hardware and software services for rehabilitation, training and activity monitoring. The rollator hardware and robotic components are provided by Inria Hephaistos). Both prototypes are being evaluated by partner medical institutions. Indes contribution to the project is supported by the EC grant and by an Inria self funded ADT (Technology Development Program).
- Hopcare. Indes collaborates with other research teams (Inria STARS, Nice University Cobtek Project) and local institutes and SMEs to foster the development distributed monitoring and supervision applications with the Hop.js technology. An expert engineer is dedicated to this project (grant from UCN@Sophia Labex, since may 2015).
  - ICP (Institut Claude Pompidou Hospital, in Nice) is now using the Alzheimer diagnosis tool developed using Hop.js. User Data generated from Inria/Stars sensors and image analysis software are collected by a Hop.js server and processed before being delivered to the Physician's web tablet, as an editable web report, or paper ready PDF reports.
  - The activity monitoring application enables real-time monitoring of various events generated by hardware/software monitoring tools (such as the video monitoring applications from Inria/Stars) as well as user defined events. Hop.js is the common framework for the whole application (communications with remote information servers, processing of input data, database management, user authentication and authorization, custom views for web clients). The application will soon be deployed at the Nice Valrose EHPAD (a specialized institution for elderly who need medical care), where Inria runs an experimentation lab.

– A third application has been developed to enable the configuration and use of Inria/Stars video analysis tools through a web interface. The application is used by researchers to tune their data processing algorithms.

9. 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, pp. 2-18

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

[3] G. BOUDOL, Z. LUO, T. REZK, M. SERRANO. *Reasoning about Web Applications: An Operational Semantics for HOP*, in "ACM Transactions on Programming Languages and Systems (TOPLAS)", 2012, vol. 34, n^{o} 2

[4] 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", 2008, pp. 323-335

[5] M. SERRANO, G. BERRY. *Multitier Programming in Hop - A first step toward programming 21st-century applications*, in "Communications of the ACM", August 2012, vol. 55, n^{o} 8, pp. 53–59 [*DOI :* 10.1145/2240236.2240253], http://cacm.acm.org/magazines/2012/8/153796-multitier-programming-in-hop/abstract

[6] 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

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

Publications of the year

Doctoral Dissertations and Habilitation Theses

[8] J. GRANDE. *Conception et implémentation d'un langage de programmation concurrente modulaire*, Université Nice Sophia Antipolis, September 2015, https://hal.inria.fr/tel-01246636

Articles in International Peer-Reviewed Journals

[9] S. CAPECCHI, I. CASTELLANI, M. DEZANI-CIANCAGLINI. *Information Flow Safety in Multiparty Sessions*, in "Mathematical Structures in Computer Science", 2015, 43 p. [*DOI :* 10.1017/S0960129514000619], https://hal.inria.fr/hal-01237236

International Conferences with Proceedings

[10] Y. COUILLEC, M. SERRANO. *Requesting Heterogeneous Data Sourceswith Array Comprehensions in Hop.js*, in "15th Symposium on Database Programming Languages", Pittsburgh, United States, ACM, October 2015, 4 p. , https://hal.inria.fr/hal-01246628

[11] J. FRAGOSO SANTOS, T. JENSEN, T. REZK, A. SCHMITT. *Hybrid Typing of Secure Information Flow in a JavaScript-like Language*, in "10th International Symposium on Trustworthy Global Computing (TGC 2015)", Madrid, Spain, August 2015, https://hal.archives-ouvertes.fr/hal-01243029

[12] J. FRAGOSO SANTOS, T. REZK, A. ALMEIDA MATOS. *Modular Monitor Extensions for Information Flow Security in JavaScript*, in "Trustworthy Global Computing", Madrid, Spain, 2015, https://hal.archives-ouvertes.fr/hal-01247123

[13] J. GRANDE, G. BOUDOL, M. SERRANO. *Jthread, a deadlock-free mutex library*, in "17th International Symposium on Principles and Practice of Declarative Programming", Sienne, Italy, July 2015, 12 p. [*DOI :* 10.1145/2790449.2790523], https://hal.inria.fr/hal-01246618

Scientific Books (or Scientific Book chapters)

[14] M. ITURBURU, E. GOIBURU, J. YANGUAS, E. ANDUEZA, E. CORRAL, C. ALDERETE, A. ORBEGOZO, D. DANEY, V. PRUNET, J.-P. MERLET. *User Needs and Requirements for the Mobility Assistance and Activity Monitoring Scenario within the RAPP Project*, in "Progress in Automation, Robotics and Measuring Techniques", R. SZEWCZYK, C. ZIELIŃSKI, M. KALICZYŃSKA (editors), Springer International Publishing, 2015, vol. 351, pp. 105-117 [*DOI :* 10.1007/978-3-319-15847-1_11], https://hal.inria.fr/hal-01145219

Research Reports

[15] B. P. SERPETTE. *Using counters for absence prediction in Esterel*, Inria Sophia Antipolis - Méditerranée, June 2015, https://hal.inria.fr/hal-01226760