IN PARTNERSHIP WITH:
**CNRS**

**Université Nantes**

**Ecole des Mines de Nantes**

# Activity Report 2014

# Project-Team ASCOLA

## Aspect and composition languages

IN COLLABORATION WITH: Laboratoire d'Informatique de Nantes Atlantique (LINA)

RESEARCH CENTER
**Rennes - Bretagne-Atlantique**

THEME
**Distributed programming and Software engineering**

# Table of contents

## Project-Team ASCOLA

**Keywords:** Software Composition, Programming Languages, Distributed Systems, Cloud Computing, Formal Methods, Energy Consumption, Security

*Creation of the Project-Team:* 2009 January 01.

# 1. Members

**Research Scientists**
Adrien Lebre [Inria, Researcher; under delegation from MN]
Nicolas Tabareau [Inria, Researcher]

**Faculty Members**
Mario Südholt [Team leader, MN, Professor, HdR]
Pierre Cointe [MN, Professor, director of the LINA research institute, HdR]
Rémi Douence [MN, Associate Professor]
Hervé Grall [MN, Associate Professor]
Thomas Ledoux [MN, Associate Professor]
Jean-Marc Menaud [MN, Professor, HdR]
Jacques Noyé [MN, Associate Professor, dean of the CS department of MN]
Flavien Quesnel [MN, until July 2014]
Jean-Claude Royer [MN, Professor, HdR]

**Engineer**
Yousri Kouki [MN, until Sep 2014]

**PhD Students**
Diana Allam [Polytech Nantes, until July 2014]
Walid Benghabrit [MN]
Paul Blouët [MN, co-supervision with Prof. De Meuter, VUB, Belgium]
Ronan-Alexandre Cherrueau [MN]
Frédéric Dumont [MN, CIFRE EasyVirt]
Simon Dupont [MN, CIFRE Sigma]
Alexandre Garnier [MN]
Guilhem Jaber [ENS Rennes, until July 2014]
Sabbir Hasan [Inria, co-supervision with Prof. Pazat, Myriads team, Inria]
Yacine Hebbal [MN, CIFRE Orange, from Oct. 2014]
Mayleen Lacouture [MN, until Oct. 2014]
Yunbo Li [Inria, co-supervision with Dr. Orgerie, Myriads team, Inria]
Florent Marchand de Kerchove de Denterghem [MN]
Jonathan Pastor [MN]
Charles Prud'homme [MN, until Feb 2014, co-supervision with Tasc team, MN]
Kevin Quirin [MN]
Jurgen Van Ham [MN, co-supervision with Prof. Mezini, TU Darmstadt, Germany]

**Post-Doctoral Fellows**
Ismael Figueroa [Inria, PhD until Apr 2014]
Hemant Kumar Mehta [Inria, until Feb 2014]
Guillaume Le Louët [MN, PhD until May 2014]
Jonathan Lejeune [MN, from Oct. 2014]
Mohamed Sellami [MN, until Aug 2014]

**Administrative Assistants**

Anne-Claire Binétruy [Inria, part time 33%, from Sep 2014]
Cecile Derouet [Inria, part time 33%, until Sep 2014]

**Other**

Gustavo Soto Ridd [Inria, Master Intership, from Aug 2014 until Nov 2014]

# 2. Overall Objectives

## 2.1. Presentation

The research team addresses the general problem of evolving software by developing concepts, languages, implementations and tools for building software architectures based on components and aspects. Its long term goal is the development of new abstractions for the programming of software architectures, their representation in terms of expressive programming languages and their correct and efficient implementation.

We pursue the following objectives:

- New concepts and techniques for the compositional definition and implementation of complex software systems, notably involving crosscutting concerns that cannot be handled modularly using traditional software development approaches.
- New programming techniques and algorithms for resource management in mutualized environments. We provide language abstractions and implementation techniques for large-scale applications in cloud- and grid-based systems, both on the level of (service-based) applications and (virtualized) infrastructures. We develop solutions, in particular, for the optimization of the energy consumption in such environments (data centers ...)
- We develop new formal theories for and apply formal methods to the correctness of software systems. We aim at developing more powerful techniques for theorem proving and enable complex, often dynamic, software systems to be proven correct using program transformations and analysis techniques. We develop solutions, in particular, for the constructive enforcement of security properties on the level of software systems.

Finally, we apply and validate our results based on real-world applications from numerous domains, notably enterprise information systems, the Cloud, and pervasive systems.

# 3. Research Program

## 3.1. Overview

Since we mainly work on new concepts for the language-based definition and implementation of complex software systems, we first briefly introduce some basic notions and problems of software components (understood in a broad sense, that is, including modules, objects, architecture description languages and services), aspects, and domain-specific languages. We conclude by presenting the main issues related to distribution and concurrency, in particular related to capacity planning issues that are relevant to our work.

## 3.2. Software Composition

**Modules and services.** The idea that building *software components*, i.e., composable prefabricated and parameterized software parts, was key to create an effective software industry was realized very early [64]. At that time, the scope of a component was limited to a single procedure. In the seventies, the growing complexity of software made it necessary to consider a new level of structuring and programming and led to the notions of information hiding, *modules*, and module interconnection languages [71], [49]. Information hiding promotes a black-box model of program development whereby a module implementation, basically a collection of procedures, is strongly encapsulated behind an interface. This makes it possible to guarantee logical invariant *properties* of the data managed by the procedures and, more generally, makes *modular reasoning* possible.

In the context of today's Internet-based information society, components and modules have given rise to *software services* whose compositions are governed by explicit *orchestration or choreography* specifications that support notions of global properties of a service-oriented architecture. These horizontal compositions have, however, to be frequently adapted dynamically. Dynamic adaptations, in particular in the context of software evolution processes, often conflict with a black-box composition model either because of the need for invasive modifications, for instance, in order to optimize resource utilization or modifications to the vertical compositions implementing the high-level services.

**Object-Oriented Programming.** Classes and objects provide another kind of software component, which makes it necessary to distinguish between *component types* (classes) and *component instances* (objects). Indeed, unlike modules, objects can be created dynamically. Although it is also possible to talk about classes in terms of interfaces and implementations, the encapsulation provided by classes is not as strong as the one provided by modules. This is because, through the use of inheritance, object-oriented languages put the emphasis on *incremental programming* to the detriment of modular programming. This introduces a white-box model of software development and more flexibility is traded for safety as demonstrated by the *fragile base class* issue [67].

**Architecture Description Languages.** The advent of distributed applications made it necessary to consider more sophisticated connections between the various building blocks of a system. The *software architecture* [75] of a software system describes the system as a composition of *components* and *connectors*, where the connectors capture the *interaction protocols* between the components [40]. It also describes the rationale behind such a given architecture, linking the properties required from the system to its implementation. *Architecture Description Languages* (ADLs) are languages that support architecture-based development [65]. A number of these languages make it possible to generate executable systems from architectural descriptions, provided implementations for the primitive components are available. However, guaranteeing that the implementation conforms to the architecture is an issue.

**Protocols.** Today, protocols constitute a frequently used means to precisely define, implement, and analyze contracts, notably concerning communication and security properties, between two or more hardware or software entities. They have been used to define interactions between communication layers, security properties of distributed communications, interactions between objects and components, and business processes.

Object interactions [69], component interactions [81], [73] and service orchestrations [50] are most frequently expressed in terms of *regular interaction protocols* that enable basic properties, such as compatibility, substitutability, and deadlocks between components to be defined in terms of basic operations and closure properties of finite-state automata. Furthermore, such properties may be analyzed automatically using, e.g., model checking techniques [47], [56].

However, the limited expressive power of regular languages has led to a number of approaches using more expressive *non-regular* interaction protocols that often provide distribution-specific abstractions, e.g., session types [58], or context-free or turing-complete expressiveness [74], [45]. While these protocol types allow conformance between components to be defined (e.g., using unbounded counters), property verification can only be performed manually or semi-automatically.

## 3.3. Programming languages for advanced modularization

The main driving force for the structuring means, such as components and modules, is the quest for clean *separation of concerns* [51] on the architectural and programming levels. It has, however, early been noted that concern separation in the presence of crosscutting functionalities requires specific language and implementation level support. Techniques of so-called *computational reflection*, for instance, Smith's 3-Lisp or Kiczales's CLOS meta-object protocol [76], [61] as well as metaprogramming techniques have been developed to cope with this problem but proven unwieldy to use and not amenable to formalization and property analysis due to their generality. Methods and techniques from two fields have been particularly useful in addressing such advanced modularization problems: Aspect-Oriented Software Development as the field concerned with the systematic handling of modularization issues and domain-specific languages that provide declarative and efficient means for the definition of crosscutting functionalities.

**Aspect-Oriented Software Development** [60], [38] has emerged over the previous decade as the domain of systematic exploration of crosscutting concerns and corresponding support throughout the software development process. The corresponding research efforts have resulted, in particular, in the recognition of *crosscutting* as a fundamental problem of virtually any large-scale application, and the definition and implementation of a large number of aspect-oriented models and languages.

However, most current aspect-oriented models, notably AspectJ [59], rely on pointcuts and advice defined in terms of individual execution events. These models are subject to serious limitations concerning the modularization of crosscutting functionalities in distributed applications, the integration of aspects with other modularization mechanisms such as components, and the provision of correctness guarantees of the resulting AO applications. They do, in particular, only permit the manipulation of distributed applications on a per-host basis, that is, without direct expression of coordination properties relating different distributed entities [77]. Similarly, current approaches for the integration of aspects and (distributed) components do not directly express interaction properties between sets of components but rather seemingly unrelated modifications to individual components [48]. Finally, current formalizations of such aspect models are formulated in terms of low-level semantic abstractions (see, e.g., Wand's et al semantics for AspectJ [80]) and provide only limited support for the analysis of fundamental aspect properties.

Different approaches have been put forward to tackle these problems, in particular, in the context of so-called *stateful* or *history-based aspect languages* [52], [53], which provide pointcut and advice languages that directly express rich relationships between execution events. Such languages have been proposed to directly express coordination and synchronization issues of distributed and concurrent applications [70], [43], [55], provide more concise formal semantics for aspects and enable analysis of their properties [41], [54], [52], [39]. Furthermore, first approaches for the definition of *aspects over protocols* have been proposed, as well as over regular structures [52] and non-regular ones [79], [68], which are helpful for the modular definition and verification of protocols over crosscutting functionalities.

They represent, however, only first results and many important questions concerning these fundamental issues remain open, in particular, concerning the semantics foundations of AOP and the analysis and enforcement of correctness properties governing its, potentially highly invasive, modifications.

**Domain-specific languages (DSLs)** represent domain knowledge in terms of suitable basic language constructs and their compositions at the language level. By trading generality for abstraction, they enable complex relationships among domain concepts to be expressed concisely and their properties to be expressed and formally analyzed. DSLs have been applied to a large number of domains; they have been particularly popular in the domain of software generation and maintenance [66], [82].

Many modularization techniques and tasks can be naturally expressed by DSLs that are either specialized with respect to the type of modularization constructs, such as a specific brand of software component, or to the compositions that are admissible in the context of an application domain that is targeted by a modular implementation. Moreover, software development and evolution processes can frequently be expressed by transformations between applications implemented using different DSLs that represent an implementation at different abstraction levels or different parts of one application.

Functionalities that crosscut a component-based application, however, complicate such a DSL-based transformational software development process. Since such functionalities belong to another domain than that captured by the components, different DSLs should be composed. Such compositions (including their syntactic expression, semantics and property analysis) have only very partially been explored until now. Furthermore, restricted composition languages and many aspect languages that only match execution events of a specific domain (e.g., specific file accesses in the case of security functionality) and trigger only domain-specific actions clearly are quite similar to DSLs but remain to be explored.

## 3.4. Distribution and Concurrency

While ASCOLA does not investigate distribution and concurrency as research domains per se (but rather from a software engineering and modularization viewpoint), there are several specific problems and corresponding approaches in these domains that are directly related to its core interests that include the structuring and modularization of large-scale distributed infrastructures and applications. These problems include crosscutting functionalities of distributed and concurrent systems, support for the evolution of distributed software systems, and correctness guarantees for the resulting software systems.

Underlying our interest in these domains is the well-known observation that large-scale distributed applications are subject to *numerous crosscutting functionalities* (such as the transactional behavior in enterprise information systems, the implementation of security policies, and fault recovery strategies). These functionalities are typically partially encapsulated in distributed infrastructures and partially handled in an ad hoc manner by using infrastructure services at the application level. Support for a more principled approach to the development and evolution of distributed software systems in the presence of crosscutting functionalities has been investigated in the field of *open adaptable middleware* [44], [63]. Open middleware design exploits the concept of reflection to provide the desired level of configurability and openness. However, these approaches are subject to several fundamental problems. One important problem is their insufficient, framework-based support that only allows partial modularization of crosscutting functionalities.

There has been some *criticism* on the use of *AspectJ-like aspect models* (which middleware aspect models like that of JBoss AOP are an instance of) for the modularization of distribution and concurrency related concerns, in particular, for transaction concerns [62] and the modularization of the distribution concern itself [77]. Both criticisms are essentially grounded in AspectJ's inability to explicitly represent sophisticated relationships between execution events in a distributed system: such aspects therefore cannot capture the semantic relationships that are essential for the corresponding concerns. History-based aspects, as those proposed by the ASCOLA project-team provide a starting point that is not subject to this problem.

From a point of view of language design and implementation, aspect languages, as well as domain specific languages for distributed and concurrent environments share many characteristics with existing distributed languages: for instance, event monitoring is fundamental for pointcut matching, different synchronization strategies and strategies for code mobility [57] may be used in actions triggered by pointcuts. However, these relationships have only been explored to a small degree. Similarly, the formal semantics and formal properties of aspect languages have not been studied yet for the distributed case and only rudimentarily for the concurrent one [41], [55].

## 3.5. Security

Security properties and policies over complex service-oriented and standalone applications become ever more important in the context of asynchronous and decentralized communicating systems. Furthermore, they constitute prime examples of crosscutting functionalities that can only be modularized in highly insufficient ways with existing programming language and service models. Security properties and related properties, such as accountability properties, are therefore very frequently awkward to express and difficult to analyze and enforce (provided they can be made explicit in the first place).

Two main issues in this space are particularly problematic from a compositional point of view. First, information flow properties of programming languages, such as flow properties of Javascript [42], and service-based systems [46] are typically specially-tailored to specific properties, as well as difficult to express and analyze. Second, the enforcement of security properties and security policies, especially accountability-related properties [72], [78], is only supported using ad hoc means with rudimentary support for property verification.

The ASCOLA team has recently started to work on providing formal methods, language support and implementation techniques for the modular definition and implementation of information flow properties as well as policy enforcement in service-oriented systems as well as, mostly object-oriented, programming languages.

## 3.6. Capacity Planning for Large Scale Distributed System

Since the last decade, cloud computing has emerged as both a new economic model for software (provision) and as flexible tools for the management of computing capacity. Nowadays, the major cloud features have become part of the mainstream (virtualization, storage and software image management) and the big market players offer effective cloud-based solutions for resource pooling. It is now possible to deploy virtual infrastructures that involve virtual machines (VMs), middleware, applications, and networks in such a simple manner that a new problem has emerged over the last two years: VM sprawl (virtual machine proliferation) that consumes valuable computing, memory, storage and energy resources, thus menacing serious resource shortages. Scientific approaches that address VM sprawl are both based on classical administration techniques like the lifecycle management of a large number of VMs as well as the arbitration and the careful management of all resources consumed and provided by the hosting infrastructure (energy, power, computing, memory, network etc.).

The ASCOLA team investigates fundamental techniques for cloud computing and capacity planning, from infrastructures to the application level. Capacity planning is the process of planning for, analyzing, sizing, managing and optimizing capacity to satisfy demand in a timely manner and at a reasonable cost. Applied to distributed systems like clouds, a capacity planning solution must mainly provide the minimal set of resources necessary for the proper execution of the applications (i.e., to ensure service level agreement, SLA). The main challenges in this context are: scalability, fault tolerance and reactivity of the solution in a large-scale distributed system, the analysis and optimization of resources to minimize the cost (mainly costs related to the energy consumption of datacenters), as well as the profiling and adaptation of applications to ensure useful levels of quality of service (throughput, response time, availability etc.).

Our solutions are mainly based on virtualized infrastructures that we apply from the IaaS to the SaaS levels. We are mainly concerned by the management and the execution of the applications by harnessing virtualization capabilities, the investigation of alternative solutions that aim at optimizing the trade-off between performance and energy costs of both applications and cloud resources, as well as arbitration policies in the cloud in the presence of energy-constrained resources.

# 4. Application Domains

## 4.1. Enterprise Information Systems and Services

Large IT infrastructures typically evolve by adding new third-party or internally-developed components, but also frequently by integrating already existing information systems. Integration frequently requires the addition of glue code that mediates between different software components and infrastructures but may also consist in more invasive modifications to implementations, in particular to implement crosscutting functionalities. In more abstract terms, enterprise information systems are subject to structuring problems involving horizontal composition (composition of top-level functionalities) as well as vertical composition (reuse and sharing of implementations among several top-level functionalities). Moreover, information systems have to be more and more dynamic.

Service-Oriented Computing (SOC) that is frequently used for solving some of the integration problems discussed above. Indeed, service-oriented computing has two main advantages:

- Loose-coupling: services are autonomous, in that they do not require other services to be executed;
- Ease of integration: Services communicate over standard protocols.

Our current work is based on the following observation: similar to other compositional structuring mechanisms, SOAs are subject to the problem of crosscutting functionalities, that is, functionalities that are scattered and tangled over large parts of the architecture and the underlying implementation. Security functionalities, such as access control and monitoring for intrusion detection, are a prime example of such a functionality in that it is not possible to modularize security issues in a well-separated module. Aspect-Oriented Software Development is precisely an application-structuring method that addresses in a systemic way the problem of the lack of modularization facilities for crosscutting functionalities.

We are considering solutions to secure SOAs by providing an aspect-oriented structuring and programming model that allows security functionalities to be modularized. Two levels of research have been identified:

- Service level: as services can be composed to build processes, aspect weaving will deal with the orchestration and the choreography of services.
- Implementation level: as services are abstractly specified, aspect weaving will require to extend service interfaces in order to describe the effects of the executed services on the sensitive resources they control.

In 2014, we have published results on constructive mechanisms for security and accountability properties in service-based system as well as results on service provisioning problems, in particular, service interoperability and mediation, see Sec. 6.3. Furthermore, we take part in the European project A4Cloud on accountability challenges, that is, the responsible stewardship of third-party data and computations, see Sec. 8.3.

## 4.2. Capacity Planning in Cluster, Grid and Cloud Computing

Cluster, Grid and more recently Cloud computing platforms aim at delivering large capacities of computing power. These capacities can be used to improve performance (for scientific applications) or availability (e.g., for Internet services hosted by datacenters). These distributed infrastructures consist of a group of coupled computers that work together and may be spread across a LAN (cluster), across a WAN (Grid), and across the Internet (Clouds). Due to their large scale, these architectures require permanent adaptation, from the application to the system level and call for automation of the corresponding adaptation processes. We focus on self-configuration and self-optimization functionalities across the whole software stack: from the lower levels (systems mechanisms such as distributed file systems for instance) to the higher ones (i.e. the applications themselves such as J2EE clustered servers or scientific grid applications).

In 2014, we have proposed a mechanism to take into account locality aspects in the DVMS proposal, a fully distributed VM scheduler. Concretely, our mechanism leverages Vivaldi coordinates in order to favor live migration of virtual machines between servers belonging to the same site before performing inter-site live migrations. By such a means, we have improved the reactivity of DVMS, establishing it as one of the most scalable and reactive scheduler of virtual machines for large-scale cloud computing infrastructures. Finally, we have also provided several results on the energy efficient management of Cloud applications and infrastructures, see Sec. 6.4.

In the energy field, we have designed a set of techniques, named Optiplace, for cloud management with flexible power models through constraint programming. OptiPlace supports external models, named views. Specifically, we have developed a power view, based on generic server models, to define and reduce the power consumption of a datacenter's physical servers. We have shown that OptiPlace behaves at least as good as our previous system, Entropy, requiring as low as half the time to find a solution for the constrained-based placement of tasks for large datacenters.

## 4.3. Pervasive Systems

Pervasive systems are another class of systems raising interesting challenges in terms of software structuring. Such systems are highly concurrent and distributed. Moreover, they assume a high-level of mobility and context-aware interactions between numerous and heterogeneous devices (laptops, PDAs, smartphones, cameras, electronic appliances...). Programming such systems requires proper support for handling various interfering concerns like software customization and evolution, security, privacy, context-awareness... Additionally, service composition occurs spontaneously at runtime.

In 2014, we have extended the language EScala, which integrates reactive programming through events with aspect-oriented and object-oriented mechanisms, see Sec. 6.3.

# 5. New Software and Platforms

## 5.1. btrCloud (and Entropy)

**Participants:** Jean-Marc Menaud [correspondent], Guillaume Le Louët, Frédéric Dumont.

Orchestration, virtualization, energy, autonomic system, placement, cloud computing, cluster, data center, scheduler, grid

btrCloud is a virtual machine manager for clusters and provides a complete solution for the management and optimization of virtualized data centers. btrCloud (acronym of better cloud) is composed of three parts.

The analysis function enables operatives and people in charge to monitor and analyze how a data-center works — be it on a daily basis, on the long run, or in order to predict future trends. This feature includes boards for performance evaluation and analysis as well as trends estimation.

btrCloud, by the integration of btrScript, provides (semi-)automated VM lifecycle management, including provisioning, resource pool management, VM tracking, cost accounting, and scheduled deprovisioning. Key features include a thin client interface, template-based provisioning, approval workflows, and policy-based VM placement.

Finally, several kinds of optimizations are currently available, such as energy and load balancing. The former can help save up to around 20% of the data-center energy consumption. The latter provides optimized quality of service properties for applications that are hosted in the virtualized datacenters.

btrCloud is available at http://www.btrcloud.org.

## 5.2. EScala and JEScala

**Participants:** Jacques Noyé [correspondent], Jurgen Van Ham.

AOP, inheritance, event-based programming, events, declarative events, asynchronous events, join operator, Scala

EScala is an extension of the programming language Scala with support for events as object members. EScala combines ideas of event-driven, aspect-oriented and functional reactive programming.

Events are natural abstractions for describing interactive behavior as part of an object interface. In conventional object-oriented languages, events are implemented indirectly, typically using the Observer pattern. C# eliminates the corresponding glue code and directly supports events as object members. However, events are still *explicitly* triggered at specific locations within the program.

EScala goes much further. First, it also supports *implicit* events. Akin to join points in aspect-oriented languages, these events are implicitly produced at specific execution points, such as the beginning or the end of the execution of a method. Second, *declarative events* make it possible to compose events using logical operators as well as to filter them and alter their content.

EScala events are fully integrated with object-oriented features. An event is defined in the context of its owner object. Event definitions are inherited in subclasses and event uses are late-bound. Unlike typical aspect-oriented languages, EScala preserves object-oriented encapsulation and modular reasoning.

JEScala extends EScala with support for concurrent programming (see Sec. 6.2). Events can be declared as *asynchronous* so that their handling takes place concurrently. A new composition operator, the *join* operator, inspired by the join calculus, can also be used to synchronize the concurrent activities created by asynchronous events and communicate between them.

This is joint work with the Software Technology Group at TU Darmstadt.

Prototype implementations of these languages are available through http://www.stg.tu-darmstadt.de/research.

## 5.3. CSLA

**Participants:** Thomas Ledoux [correspondent], Yousri Kouki.

Service-level agreement, Cloud computing, elasticity

Verifying non-functional properties like performance, dependability, energy consumption and economical costs of Cloud is challenging today due to ad hoc management in terms of Quality-of-Service (QoS). We believe that a differentiating element between Cloud computing environments will be the QoS and the Service-Level Agreement (SLA) provided by the Cloud.

CSLA, the Cloud Service Level Agreement language, allows the definition of SLA properties for arbitrary Cloud services (XaaS). CSLA addresses QoS uncertainty in unpredictable and dynamic environment and provides a cost model of Cloud computing. Besides the standard formal definition of contracts – comprising validity, parties, services definition and guarantees/violations – CSLA is enriched with features, such as QoS degradation and an advanced penalty model, thus introducing fine-grained language support for Cloud elasticity management [27][26].

CSLA is available at http://www.emn.fr/z-info/csla.

## 5.4. SAdapt

**Participants:** Ronan-Alexandre Cherrueau [correspondent], Mario Südholt.

Service-oriented systems, distributed programming, event-based programming, workflow patterns

The SAdapt tool provides an implementation of workflow adaptation patterns and allows the transformation of service-oriented systems implemented using Apache's CXF service infrastructure in terms of high-level declarative service transformations. The transformations are defined using an expressive language that supports matching of the execution of service-based systems in terms of flexible patterns over service compositions.

The SAdapt tool has partially been developed and is employed in the A4Cloud EU project (see Sec. 8.3) as a basis for our work on the enforcement of accountability properties in complex cloud-based systems.

The SAdapt tool and its application, notably to the security hardening of service systems that use OAuth 2 for the authorization of resource accesses is available at http://a4cloud.gforge.inria.fr/doku.php?id=start:advservcomp.

In 2014, we have used and extended the tool in order to investigate accountability properties of service-based applications, see Sec. 6.3.

## 5.5. SimGrid/VMPlaces

**Participants:** Takahiro Hirofuchi, Adrien Lebre [correspondent], Jonathan Pastor, Flavien Quesnel, Mario Südholt.

Simulation, Virtualization, Cloud computing, VM placement

SimGrid is a toolkit for the simulation of algorithms executed on large-scale distributed systems. Developed for more than a decade, it has been used in a large number of studies described in more than 100 publications. In 2013, ASCOLA with the support of the SimGrid core-developers, designed and implemented additional capabilities, in particular the Virtual Machine abstraction, enabling to address Cloud Computing related concerns.

Developed, first, in an experimental repository, the integration of these extensions into the master branch of SimGrid has been achieved during Summer 2014. The principal role of ASCOLA is now to ensure the maintenance of this portion of the code with respect to the evolutions of the SimGrid toolkit (such as for instance the recent port of the SimGrid kernel in C++).

Although the virtualization extensions are recent, several projects leveraging them have been already proposed. [1] Among them, ASCOLA is working on dedicated framework to evaluate and compare VM placement algorithms. Entitled VMPlaces, this framework is composed of two major components: the injector and the VM placement algorithm. The injector is the generic part of the framework (i.e. the one you can directly use) while the VM placement algorithm is the part you want to study (or compare with available algorithms). Currently, the VMPlaceS is released with three algorithms:

- Entropy, a centralized approach using a constraint programming approach to solve the placement/reconfiguration VM problem
- Snooze, a hierarchical approach where each manager of a group invokes Entropy to solve the placement/reconfiguration VM problem. Note that in the original implementation of Snooze, it is using a specific heuristic to solve the placement/reconfiguration VM problem. As the sake of simplicity, we have simply reused the entropy scheduling code.
- DVMS, a distributed approach that dynamically partitions the system and invokes Entropy on each partition.

SimGrid is available at http://simgrid.gforge.inria.fr.
VMPlaces is available at http://beyondtheclouds.github.io/VMPlaceS/

# 6. New Results

## 6.1. Highlights of the Year

Nicolas Tabareau was awarded a starting grant from the European Research Council (ERC), the most prestigious type of research projects of the European Union for young researchers. From 2015–2020 he will pursue research on “CoqHoTT: Coq for Homotopy Type Theory.”

Jonathan Pastor has won the joint 1st prize at the Grid5000 Scale challenge, an international challenge for large-scale experiments on geographically-distributed cluster environments. Jonathan has shown with a colleague how to deploy and manage thousands of VMs in such an environment using his approach to fully distributed virtual machine management.

This year we have provided major research results in two domains. First, we have developed several new approaches for the formal reasoning over software in the domains of theorem proving [31], as well as reasoning over distributed interaction protocols [32] and software compositions [24]. Second, we have developed new methods supporting dynamic computations over the cloud, both by means of more elastic cloud applications [27] and better locality management for the dynamic placement of virtual machines in Cloud infrastructures [29].

## 6.2. Programming Languages

**Participants:** Ronan-Alexandre Cherrueau, Rémi Douence, Hervé Grall, Thomas Ledoux, Florent Marchand de Kerchove de Denterghem, Jacques Noyé, Jean-Claude Royer, Mario Südholt.

### *6.2.1. Formal Methods, logics and type theory*

This year we have published new results extending previous type theories: we have introduced a notion of universe polymorphism for the theorem prover Coq and new type-based mechanisms for the definition and analysis of program equivalences. We have also shown how to harness capabilities, well-known in the security domain, in the context of the functional programming language Haskell. These results are detailed in the current section.

[1] The list of the projects is available at : http://simgrid.gforge.inria.fr/contrib/clouds-sg-doc.html

Furthermore, we have applied formal methods and typing in the context of aspect oriented programming ([12], [16], [24]) and in the context of distributed programming (aspectual session types [32]). We have also developed a framework for the formal definition and analysis of accountability properties based on temporal logics. These different results are detailed in Sec. 6.3 for details.

#### *6.2.1.1. Universe Polymorphism in Coq*

Universes are used in type theory to ensure consistency by checking that definitions are well-stratified according to a certain hierarchy. In the case of the Coq proof assistant, based on the predicative Calculus of Inductive Constructions (pCIC), this hierarchy is built from an impredicative sort Prop and an infinite number of predicative Type universes. A cumulativity relation represents the inclusion order of universes in the core theory. Originally, universes were thought to be floating levels, and definitions to implicitly constrain these levels in a consistent manner. This works well for most theories, however the globality of levels and constraints precludes generic constructions on universes that could work at different levels. We have introduced universe polymorphism [31] that extends this setup by adding local bindings of universes and constraints, supporting generic definitions over universes, reusable at different levels. This provides the same kind of code reuse facilities as ML-style parametric polymorphism. However, the structure and hierarchy of universes is more complex than bare polymorphic type variables.

#### *6.2.1.2. A Logical Study of Program Equivalence*

Proving program equivalence for a functional language with references is a notoriously difficult problem. The goal of the thesis of Guilhem Jaber on “A Logical Study of Program Equivalence” [G. Jaber, Mines Nantes, July 14] was to propose a logical system in which such proofs can be formalized, and in some cases inferred automatically. In the first part, a generic extension method of dependent type theory has been proposed, based on a forcing interpretation seen as a presheaf translation of type theory. This extension equips type theory with guarded recursive constructions, which are subsequently used to reason on higher-order references. In the second part, he has defined a nominal game semantics for a language with higher-order references. It marries the categorical structure of game semantics with a trace representation of denotations of programs, which can be computed operationally and thus have good modularity properties. Using this semantics, he has proven completeness of Kripke logical relations defined in a direct way, using guarded recursive types, without using biorthogonality. The problem of contextual equivalence is then reduced to the satisfiability of an automatically generated formula defined in this logic, that is, to the existence of a world validating this formula. Under some conditions, this satisfiability can be decided using a SMT solver.

#### *6.2.1.3. Effect Capabilities For Haskell*

Computational effects complicate the tasks of reasoning about and maintaining software, due to the many kinds of interferences that can occur. While different proposals have been formulated to alleviate the fragility and burden of dealing with specific effects, such as state or exceptions, there is no prevalent robust mechanism that addresses the general interference issue. Building upon the idea of capability-based security, we have proposed effect capabilities [25] as an effective and flexible manner to control monadic effects and their interferences. Capabilities can be selectively shared between modules to establish secure effect-centric coordination. We have further refined capabilities with type-based permission lattices to allow fine-grained decomposition of authority. An implementation of effect capabilities in Haskell has been done, using type classes to establish a way to statically share capabilities between modules, as well as to check proper access permissions to effects at compile time.

### *6.2.2. Language Mechanisms*

In 2014, we have proposed new general language-based mechanisms for concurrent event-based systems and sequential programming languages. Moreover, we have investigated domain-specific languages that support aspect-oriented programming and provide control over propagation strategies in constraint solvers. These results are detailed in the remainder of this section.

Furthermore, we have proposed language support for the definition and enforcement of security properties, in particular related to the accountability of service-based systems, see Sec. 6.3.

#### *6.2.2.1. Concurrent Event-Based Programming*

Advanced concurrency abstractions overcome the drawbacks of low-level techniques such as locks and monitors, freeing programmers that implement concurrent applications from the burden of concentrating on low-level details. However, with current approaches the coordination logic involved in complex coordination schemas is fragmented into several pieces including join patterns, data emissions triggered in different places of the application, and the application logic that implicitly creates dependencies among communication channels, hence indirectly among join patterns. In [33], we have presented JEScala, a language that captures coordination schemas in a more expressive and modular way by leveraging a seamless integration of an advanced event system with join abstractions. We have validated the approach with case studies and provided a first performance assessment.

#### *6.2.2.2. Lazy imperative programming*

Laziness is a powerful concept in functional programming that permits the reuse of general functions in a specific context, while keeping performance close to the efficiency of dedicated definitions. Lazy evaluation can be used in imperative programming too. Twenty years ago, John Launchbury was already advocating for lazy imperative programming, but the level of laziness of his framework remained limited. Twenty years after, the picture has not changed.

We have proposed an Haskell framework to specify computational effects of imperative programs as well as their dependencies [23]. We have presented a semantics of a call-by-need lambda-calculus extended with imperative strict and lazy features and proved the correctness of our approach. While originally motivated by a less rigid use of foreign functions, we have shown that our approach is fruitful for a simple scenario based on sorted mutable arrays. Furthermore, we can take advantage of equations between algebraic operations to dynamically optimize compositions of imperative computations.

#### *6.2.2.3. Domain-Specific Aspect Languages*

Domain-Specific Aspect Languages (DSALs) are Domain-Specific Languages (DSLs) designed to express crosscutting concerns. Compared to DSLs, their aspectual nature greatly amplifies the language design space. In the context of the Associate Team RAPIDS/REAL, we have structured this space in order to shed light on and compare the different domain-specific approaches to deal with crosscutting concerns [37]. We have reported on a corpus of 36 DSALs covering the space, discussed a set of design considerations and provided a taxonomy of DSAL implementation approaches. This work serves as a frame of reference to DSAL and DSL researchers, enabling further advances in the field, and to developers as a guide for DSAL implementations.

#### *6.2.2.4. Controlling constraint propagation*

Constraint propagation is at the heart of constraint solvers. Two main trends co-exist for its implementation: variable-oriented propagation engines and constraint-oriented propagation engines. These two approaches ensure the same level of local consistency but their efficiency (computation time) can be quite different depending on the problem instances to be solved. However, it is usually accepted that there is no best approach in general, and modern constraint solvers implement only one of them.

In the context of Charles Prud'homme's PhD Thesis [15], we have gone a step further providing a solver independent language at the modeling stage to enable the design of propagation engines. We have validated our proposal with a reference implementation based on the Choco solver and the MiniZinc constraint modeling language.

## 6.3. Software Composition

**Participants:** Diana Allam, Walid Benghabrit, Ronan-Alexandre Cherrueau, Rémi Douence, Hervé Grall, Thomas Ledoux, Jean-Claude Royer, Mohamed Sellami, Mario Südholt.

### *6.3.1. Constructive Security*

Nowadays we are witnessing the wide-spread use of cloud services. As a result, more and more end-users (individuals and businesses) are using these services for achieving their electronic transactions (shopping, administrative procedures, B2B transactions, etc.). In such scenarios, personal data is generally flowing between several entities and end-users need (i) to be aware of the management, processing, storage and retention of personal data, and (ii) to have necessary means to hold service providers accountable for the usage of their data. Usual preventive security mechanisms are not adequate in a world where personal data can be exchanged on-line between different parties and/or stored at multiple jurisdictions. Accountability becomes a necessary principle for the trustworthiness of open computer systems. It regards the responsibility and liability for the data handling performed by a computer system on behalf of an organization. In case of misconduct (e.g. security breaches, personal data leak, etc.), accountability should imply remediation and redress actions, as in the real life.

In 2014, we have developed two general approaches for the definition and enforcement of accountability properties.

#### *6.3.1.1. Logic-based accountability properties*

We have proposed a framework for the representation of cloud accountability policies [19]. Such policies offer end-users a clear view of the privacy and accountability obligations asserted by the entities they interact with, as well as means to represent their preferences. This framework comes with two novel accountability policy languages; an abstract one, which is devoted for the representation of preferences/obligations in an human readable fashion, a concrete one for the mapping to concrete enforceable policies. We motivate our solution with concrete use case scenarios. [30] discusses issues related to data privacy and big data technologies and advocate the use of the framework to support accountability.

We have provided an abstract language for the representation of accountability obligations [20]. We define its semantics using first-order temporal logic and a specific modality for accountability is introduced. We analyze a healthcare use case to illustrate the efficiency of our approach in representing accountability obligations in realistic situations. The use of such services-based applications usually implies the flow of personal data on-line between several parties. In [21], we consider this issue at the design-time of the software and we propose some foundations for an accountable software design. Accountability for a software is a property describing, among other aspects, its liability to end-users for the usage of the data it has been entrusted. We propose to enrich software's component design by accountability clauses using an abstract accountability language (introduced in [20]). We also define conditions for the well-formedness of an accountable component design and show how they can be checked using the $\mu$-CRL model-checker.

#### *6.3.1.2. Defining and enforcing multi-level accountability properties*

Many accountability policies require access to all levels of the software stack of service-based applications. Furthermore, they should include explicit means for the definition of cross-domain policies and provide constructive means for the implementation of a wide variety of of accountability properties. These features, in particular, multi-level support, are missing in existing approaches.

We have provided an approach that addresses these objectives explicitly through a language for the definition of expressive regular policies over accountability predicates applicable at all levels of the service stack [22]. Furthermore, we have presented hierarchies of constructive schemes for the implementation of policies for transparency and remediation properties that are implemented in terms of our accountability policy language. Finally, we have shown how to harness the accountability schemes to tackle real-world violations of accountability properties arising from security vulnerabilities of OAuth-based authorization and authentication protocols.

### *6.3.2. Aspect-Oriented Programming*

We have produced in 2014 a range of results enabling reasoning over aspect languages and investigated the use of execution levels. These results are presented in the remainder of this section.

We have also applied ideas from aspect oriented programming in the context of distributed programming (aspectual session types [32]), see Sec. 6.4.

#### *6.3.2.1. Reasoning about aspect interference using effective aspects*

Aspect-oriented programming (AOP) aims at enhancing modularity and reusability in software systems by offering an abstraction mechanism to deal with crosscutting concerns. But, in most general-purpose aspect languages aspects have almost unrestricted power, eventually conflicting with these goals. To tame aspects, we have proposed Effective Aspects: a novel approach to embed the pointcut/advice model of AOP in a statically-typed functional programming language like Haskell; along two main contributions. First, we have defined a monadic embedding of the full pointcut/advice model of AOP [16].

Type soundness is guaranteed by exploiting the underlying type system, in particular phantom types and a new anti-unification type class. In this model aspects are first-class, can be deployed dynamically, and the pointcut language is extensible, therefore combining the flexibility of dynamically-typed aspect languages with the guarantees of a static type system. Monads (which allow the definition of sequences of computations in functional programs) enable us to directly reason about computational effects both in aspects and base programs using traditional monadic techniques. Using this we extend the notion of Open Modules with effects, and also with protected pointcut interfaces to external advising. These restrictions are enforced statically using the type system. Also, we adapt the techniques of EffectiveAdvice to reason about and enforce control flow properties as well as to control effect interference. We show that the parametricity-based approach to effect interference falls short in the presence of multiple aspects and propose a different approach using monad views, a novel technique for handling the monad stack, developed by Schrijvers and Oliveira. Then, we exploit the properties of our model to enable the modular construction of new semantics for aspect scoping and weaving. Our second contribution [24] builds upon a powerful model to reason about mixin-based composition of effectful components and their interference, based on equational reasoning, parametricity, and algebraic laws about monadic effects. Our contribution is to show how to reason about interference in the presence of unrestricted quantification through pointcuts. We show that global reasoning can be compositional, which is key for the scalability of the approach in the face of large and evolving systems. A comprehensive version of those two works appears in Ismael Figueroa PhD thesis [12].

#### *6.3.2.2. Execution Levels for AOP: from program design to applications*

In AOP languages, advice evaluation is usually considered as part of the base program evaluation. This is also the case for certain pointcuts, such as if pointcuts in AspectJ, or simply all pointcuts in higher-order aspect languages like AspectScheme. While viewing aspects as part of base level computation clearly distinguishes AOP from reflection, it also comes at a price: because aspects observe base level computation, evaluating pointcuts and advice at the base level can trigger infinite regression. To avoid these pitfalls, aspect languages propose ad hoc mechanisms, which increase the complexity for programmers while being insufficient in many cases. We have proposed to clarify the situation by introducing levels of execution in the programming language [18], thereby allowing aspects to observe and run at specific, possibly different, levels. We have adopted a defensive default that avoids infinite regression, and gives advanced programmers the means to override this default using level-shifting operators.

### *6.3.3. Service provisioning*

This year, we have provided results on two fundamental problems of service-oriented architectures: service interoperability and service mediation.

#### *6.3.3.1. Service interoperability*

Web service support a document-oriented style for clients to interact with a server and promote an environment for systems that is loosely coupled and interoperable. Two models exist for implementing Web services: A process-oriented Web services model, SOAP, and a resource-oriented Web services model, RESTful. Service components are mainly based on description interfaces. These interfaces are often known as structural standardized interfaces like WSDL for SOAP and WADL for RESTful. The implementation of Web services is increasingly based on object-oriented (OO) frameworks, at the client and the server sides. Using these frameworks, developers can transform an object code into a Web service, or access a remote Web service, at the touch of a button. In this context, two levels are present: an object level built over a service level.

Diana Allam's PhD thesis [11] has focused on two properties of these frameworks:

- The loose coupling between the two levels, which allows the complex technical details of the service level to be hidden at the object level and the service level to be evolved with a minimal impact on the object level.
- The interoperability induced by the substitution principle associated to subtyping in the object level, which allows to freely convert a value of a subtype into a value of a supertype.

The thesis provides three contributions in this context. We propose a unified formal model for web services based on message passing and enabling first class channels. It is equipped with a powerful type-checking allowing union, intersection and negation operations as well as subtyping. The type checking algorithm relies on the semantic approach defined by G. Castagna. This type system is also protected against attackers. The second contribution is a concrete refinement of the model into RESTful and SOAP frameworks as well as a unified API for service discovery. To define such an API, we have first shown how the details of the standard interfaces (WSDL and WADL) could be simplified and abstracted and then we rely on subtyping in the discovery mechanism. Finally, to solve some of the interoperability issues between the OO level and the service level a formalization of the binding using categorical concepts (commutative diagrams) is proposed. Based on this an analysis of the mismatch problems has been done and a new specification of the data binding has been formalized. The document then discusses some variations in the implementation of the data binding solution and a prototype for the Apache CXF framework.

Mayleen Lacouture's PhD thesis "A Chemical Programming Language for Orchestrating Services - Application to Interoperability Problems" [M. Lacouture, MN/U. Nantes, Oct. 14] proposes a framework easing interoperability in the form of an architecture that integrates different orchestration languages with heterogeneous service providers around a pivot language. The pivot language is implemented as a new orchestration language based on the chemical programming paradigm. Concretely, the dissertation presents a language called Criojo that implements and extends the Heta-calculus, an original calculus associated to a chemical abstract machine dedicated to service-oriented computing. The consequence of adopting this approach would be an improvement in the interoperability of services and orchestration languages, thus easing the development of composite services. The high level of abstraction of Criojo could allow developers to write very concise orchestrations since message exchanges are represented in a natural and intuitive way.

#### *6.3.3.2. Service mediation*

Service composition is a major advance service-oriented computing brings to enable the development of distributed applications. However, the distributed nature of services hampers their composition with data heterogeneity problems. We address these problems with a decentralized Mediation-as-a-Service architecture that solves data inconsistencies occurring during the composition of business services [17]. As an extension to our previous work that focused on data interpretation problems, we present in this paper a solution to solve data inconsistencies at the syntactic, structural and semantic levels. We show how syntactic, structural and semantic mediation techniques can be combined, and how semantic mediation provides useful information that helps structural and syntactic mediation. We demonstrate how our architecture enables decentralized publication and discovery of mediation services. We motivate our work with a concrete scenario and validate our proposal with experiments.

### *6.3.4. Software product line architectures*

Software product lines were designed from the product line tested out by H. Ford at the beginning of the 20TH century, which led to the success of his automotive production. For 15 years, these methods have been visible in several software application fields: telephony at Nokia, televisions at Philips, print software at HP and flight applications at Boeing, among others. The concept of architecture is crucial for classic software applications, and this concept is even more important at the level of domain engineering in product lines. In a product line, the so-called reference architecture generically describes the architectures of all the products in the family. The chapter [34] describes the technical means and methods for defining a reference architecture for a software product line. It also presents the methods for operating this architecture through, for example, techniques emerging from model and software component engineering, or aspect-oriented programming. These concepts and techniques are illustrated using a case study.

## 6.4. Cloud applications and infrastructures

**Participants:** Adrien Lebre, Thomas Ledoux, Yousri Kouki, Guillaume Le Louët, Jean-Marc Menaud, Jonathan Pastor, Flavien Quesnel, Mario Südholt.

In 2014, we have provided solutions for Cloud-based and distributed programming, virtual environments and data centers, in particular concerning energy-optimal Cloud applications.

### *6.4.1. Cloud and distributed programming*

This year we have published results on a broker that provides better guarantees on service-level agreements in the Cloud. Furthermore, we have extended a class of formally-defined protocols, session types.

#### *6.4.1.1. Service-level agreement for the Cloud*

Elasticity is the intrinsic element that differentiates Cloud Computing from traditional computing paradigms, since it allows service providers to rapidly adjust their needs for resources to absorb the demand and hence guarantee a minimum level of Quality of Service (QoS) that respects the Service Level Agreements (SLAs) previously defined with their clients. However, due to non-negligible resource initiation time, network fluctuations or unpredictable workload, it becomes hard to guarantee QoS levels and SLA violations may occur.

We propose a language support for Cloud elasticity management that relies on CSLA (Cloud Service Level Agreement) [27]. CSLA offers new features such as QoS/functionality degradation and an advanced penalty model that allow providers to finely express contracts so that services self-adaptation capabilities are improved and SLA violations minimized. The approach was evaluated with a real infrastructure and application testbed. Experimental results show that the use of CSLA makes Cloud services capable of absorbing more peaks and oscillations by trading-off the QoS levels and costs due to penalties.

#### *6.4.1.2. AO session types for distributed protocols*

Multiparty session types allow the definition of distributed processes with strong communication safety properties. A global type is a choreographic specification of the interactions between peers, which is then projected locally in each peer. Well-typed processes behave accordingly to the global protocol specification. Multiparty session types are however monolithic entities that are not amenable to modular extensions. Also, session types impose conservative requirements to prevent any race condition, which prohibit the uniform application of extensions at different points in a protocol. We have proposed a means to support modular extensions with aspectual session types [32], a static pointcut/advice mechanism at the session type level. To support the modular definition of crosscutting concerns, we have augmented the expressivity of session types to allow harmless race conditions. As a result, aspectual session types make multiparty session types more flexible, modular, and extensible.

### *6.4.2. Virtualization and data centers*

In 2014, we have produced a variety of results on a new model for utility computing that addresses fundamental shortcomings of today's Cloud computing model. Furthermore, we have provided more powerful techniques for the virtualization of computations and the management of cluster-based environments, such as data centers.

#### *6.4.2.1. Next generation utility computing*

To accommodate the ever-increasing demand for Utility Computing (UC) resources while taking into account both energy and economical issues, the current trend consists in building larger and larger data centers in a few strategic locations. Although such an approach enables to cope with the actual demand while continuing to operate UC resources through centralized software system, it is far from delivering sustainable and efficient

UC infrastructures. Throughout the Discovery initiative [2], we investigate how UC resources can be managed differently, considering locality as a primary concern. Concretely, we study how it can be possible to leverage any facilities available through the Internet in order to deliver widely distributed UC platforms that can better match the geographical dispersal of users as well as the unending resource demand. Critical to the emergence of such locality-based UC (LUC) platforms is the availability of appropriate operating mechanisms. We presented a prospective vision of a unified system driving the use of resources at an unprecedented scale by turning a complex and diverse infra structure into a collection of abstracted computing facilities that is both easy to operate and reliable [35]. By deploying and using such a LUC Operating System on backbones, our ultimate vision is to make possible to host/operate a large part of the Internet by its internal structure itself: A scalable and nearly infinite set of resources delivered by any computing facilities forming the Internet, starting from the larger hubs operated by ISPs, governments and academic institutions to any idle resources that may be provided by end-users. We highlight that this work is conducted through a collaboration between the ASAP, ASCOLA, AVALON and MYRIADS Inria Project-teams.

*6.4.2.2. Adding locality capabilities to virtual machine schedulers*

Through the DVMS proposal, we showed in 2013 the benefit of leveraging peer-to-peer algorithms to design and implement virtual machines (VMs) scheduling algorithms. Although P2P based proposals considerably improve the scalability, leading to the management of hundreds of thousands of VMs over thousands of physical machines (PMs), they do not consider the network overhead introduced by multi-site infrastructures. This over- head can have a dramatic impact on the performance if there is no mechanism favoring intra-site v.s. inter-site manipulations. This year, we extended our DVMS mechanism with a new building block designed on top of the Vivaldi coordinates mechanism. We showed its benefits by discussing several experiments performed on four distinct sites of the Grid'5000 testbed. With our proposal and without changing the scheduling decision algorithm, the number of inter-site operations has been reduced by 72% [29]. This result provides a glimpse of the promising future of using locality properties to improve the performance of massive distributed Cloud platforms. We highlight that this work has been performed in collaboration with the ASAP, ASCOLA, AVALON and MYRIADS Inria Project-teams.

*6.4.2.3. WAN-wide elasticity capabilities for distributed file systems*

Applications dealing with huge amounts of data suffer significant performance impacts when they are deployed on top of an hybrid platform (i.e the extension of a local infrastructure with external cloud resources). More precisely, through a set of preliminary experiments we shew that mechanisms which enable on demand extensions of current Distributed File Systems (DFSes) are required. These mechanisms should be able to leverage external storage resources while taking into account the performance constraints imposed by the physical network topology used to interconnect the different sites. To address such a challenge we presented the premises of the Group Based File System, a glue providing the elasticity capability for storage resources by federating on demand any POSIX file systems [28].

### *6.4.3. Energy optimization*

Demand for Green services is increasing considerably as people are getting more environmental conscious to build a sustainable society. Therefore, enterprise and clients want to shift their workloads towards green Cloud environment offered by the Infrastructure-as-a-Service (IaaS) provider. The main challenge for an IaaS provider is to determine the best trade-off between its profit while using renewable energy and customers satisfaction. In order to address this issue, we propose a *Cloud energy broker* [26], which can adjust the availability and price combination to buy Green energy dynamically from the market to make datacenter green. Our energy broker tries to maximize of using renewable energy under strict budget constraint whereas it also tries to minimize the use of brown energy by capping the limit of overall energy consumption of datacenter. The energy broker was evaluated with a real workload traced by PlanetLab. Experimental results show that our energy broker successfully enables meeting the best trade-off.

[2] http://beyondtheclouds.github.io

# 7. Bilateral Contracts and Grants with Industry

## 7.1. Cooperation with SIGMA group

**Participants:** Thomas Ledoux [correspondent], Simon Dupont.

In 2012, we have started a cooperation with Sigma Group (http://www.sigma.fr), a software editor and consulting enterprise. The cooperation consists in a joint (a so-called Cifre) PhD on eco-elasticity of software for the Cloud and the sponsorship of several engineering students at the MSc-level.

As a direct consequence of the increasing popularity of Cloud computing solutions, data centers are rapidly growing in number and size and have to urgently face with energy consumption issues. The aim of Simon Dupont's PhD, started in November 2012, is to explore the *software elasticity* capability in Software-as-a-Service (SaaS) development to promote the management of SaaS applications that are more flexible, more reactive to environment changes and therefore self-adaptive for a wider range of contexts. As a result, SaaS applications become more elastic and by transitivity more susceptible to energy constraints and optimization issues.

In 2014, we have performed real world evaluations within Sigma's data centers that validated the results on new techniques for the management of elasticity within Cloud applications [27]. We have also presented our current work at [3].

# 8. Partnerships and Cooperations

## 8.1. Regional Initiatives

### *8.1.1. Competitiveness cluster Images-et-Reseaux*

#### *8.1.1.1. EcoCloud*

**Participant:** Jean-Marc Menaud.

The project EcoCloud is a cooperative research project running for 2 years. Three other partners collaborate within the project that is coordinated by the company EasyVirt: the Ascola team and another company Pentasonic. The partners aim at developing an economically-valid and ecologic cloud platform in the context of micro and mono-site data centers (all resources are in the same physical location). A high SLA level must be provided with a specific focus on high availability satisfying strong redundancy and placement constraints.

## 8.2. National Initiatives

### *8.2.1. CominLabs laboratory of excellence*

#### *8.2.1.1. EPOC*

**Participants:** Jean-Marc Menaud [coordinator], Thomas Ledoux.

The project EPOC (Energy Proportional and Opportunistic Computing system) is an (academic) Labex CominLabs project running for 4 years. Four other partners collaborate within the project that is coordinated by ASCOLA: Myriads team, and the three institutions ENIB, ENSTB and University of Nantes. In this project, the partners focus on energy-aware task execution from the hardware to application's components in the context of a *mono-site* data center (all resources are in the same physical location) which is connected to the *regular electric Grid and to renewable energy sources* (such as windmills or solar cells). Three major challenges are addressed in this context: Optimize the energy consumption of distributed infrastructures and service compositions in the presence of ever more dynamic service applications and ever more stringent availability requirements for services; Design a clever cloud's resource management which takes advantage of renewable energy availability to perform opportunistic tasks, then exploring the trade-off between energy saving and performance aspects in large-scale distributed system; Investigate energy-aware optical ultra high-speed interconnection networks to exchange large volumes of data (VM memory and storage) over very short periods of time.

[3]GreenTouch @ Nantes Digital Week

One of the strengths of the project is to provide a systematic approach, and use a single model for the system (from hard to soft) by mixing constraint programming and behavioral models to manage energy consumption in data centers.

This year, we have proposed a Cloud energy broker [26], which can adjust the availability and price combination to buy Green energy dynamically from the market to make datacenter green.

#### *8.2.1.2. SecCloud*

**Participants:** Jacques Noyé [coordinator], Florent Marchand de Kerchove de Denterghem, Mario Südholt.

The high-level objective of the 3-year SecCloud (Secure Scripting for the Cloud) project is to enhance the security of devices on which web applications can be downloaded, i.e. to enhance client-side security in the context of the Cloud. In order to do so, the project relies on a language-based approach, focusing on three related issues:

- The definition of security policies for web architectures, especially on the client-side.
- Formally-proven analyses of web programming languages.
- Multi-level enforcement mechanisms for the security policies (based on static and dynamic analysis encompassing application-level and system-level software).

ASCOLA members are mainly interested in JavaScript as a programming language as well as the use of aspects as a seamless path from the definition of security policies and their composition to their implementation.

This year we have investigated how to extend real-world Javascript environments, such as Narcissus in a modular way.

### *8.2.2. ANR*

#### *8.2.2.1. MyCloud (ANR/ARPEGE)*

**Participants:** Thomas Ledoux [coordinator], Jean-Marc Menaud, Yousri Kouki.

The MyCloud project is an ANR/ARPEGE project running for 42 months, starting in Nov. 2010. It was accepted in Jul. 2010 for funding amounting to 190 KEUR (ASCOLA only). MyCloud involves a consortium with three academic partners (Inria, LIP6, EMN) and one industrial partner (We Are Cloud).

Cloud Computing provides a convenient means of remote on-demand and pay-per-use access to computing resources. However, its ad-hoc management of quality-of-service (QoS) and SLA poses significant challenges to the performance, dependability and costs of online cloud services.

The objective of MyCloud (http://mycloud.inrialpes.fr) is to define and implement a novel cloud model: SLAaaS (SLA as a Service). The SLAaaS model enriches the general paradigm of Cloud Computing and enables systematic and transparent integration of SLA to the cloud. From the cloud provider's point of view, MyCloud proposes autonomic SLA management to handle performance, availability, energy and cost issues in the cloud. From the cloud customer's point of view, MyCloud provides SLA governance allowing cloud customers to be part of the loop and to be automatically notified about the state of the cloud, such as SLA violation and cloud energy consumption.

The project ended in April 2014. This year, our main contribution is a new system for the specification of service-level agreements in the Cloud presented at the IEEE/ACM CCGrid conference [27].

#### *8.2.2.2. SONGS (ANR/INFRA)*

**Participants:** Adrien Lebre [coordinator], Flavien Quesnel, Jonathan Pastor.

The SONGS project (Simulation of Next Generation Systems) is an ANR/INFRA project running for 48 months (starting in January 2012 with an allocated budget of 1.8MEuro, 95KEuro for ASCOLA).

The consortium is composed of 11 academic partners from Nancy (AlGorille, coordinator), Grenoble (MESCAL), Villeurbanne (IN2P3 Computing Center, GRAAL/Avalon - LIP), Bordeaux (CEPAGE, HiePACS, RUNTIME), Strasbourg (ICPS - LSIIT), Nantes (ASCOLA), Nice (MASCOTTE, MODALIS).

The goal of the SONGS project (http://infra-songs.gforge.inria.fr) is to extend the applicability of the SimGrid simulation framework from Grids and Peer-to-Peer systems to Clouds and High Performance Computation systems.

### 8.2.3. FSN

#### 8.2.3.1. OpenCloudware (FSN)

**Participants:** Jean-Marc Menaud [coordinator], Thomas Ledoux, Yousri Kouki.

The OpenCloudware project is coordinated by France Telecom, funded by the French Fonds National pour la Société Numérique (FSN, call Cloud n°1) and endorsed by competitiveness clusters Minalogic, Systematic and SCS. OpenCloudware is developed by a consortium of 18 partners bringing together industry and academic leaders, innovative technology start-ups and open source community expertise. Duration: 36 months - 2012–2014.

The OpenCloudware project aims at building an open software engineering platform, for the collaborative development of distributed applications to be deployed on multiple Cloud infrastructures. It will be available through a self-service portal. We target virtualized multi-tier applications such as JavaEE - OSGi. The results of OpenCloudware will contain a set of software components to manage the lifecycle of such applications, from modelling(Think), developing and building images (Build), to a multi-IaaS compliant PaaS platform (Run).

The ASCOLA project-team is mainly involved in the sub-projects "Think" (SLA model across Cloud layers) and "Run" (virtual machine manager for datacenters and placement constraints). In 2013, the team has developed btrCloudStack, a private cloud based on the OpenSource CloudStack and integrating the work on placement rules and energy optimization.

## 8.3. European Initiatives

### 8.3.1. FP7 & H2020 Projects

#### 8.3.1.1. ERC Starting Grant: The CoqHoTT project

**Participant:** Nicolas Tabareau [coordinator].

CoqHoTT stands for Coq for Homotopy Type Theory. The goal of this project is to go further in the correspondence between proofs and programs which has allowed in the last 20 years the development of useful proof assistants, such as Coq (developed by Inria). This project starts from the recent discovery by field medal Vladimir Voevosdky, of the strong link between homotopy theory (which studies the notion of continuous deformation in topology) and type theory (which is at the heart of the Coq proof assistant). The main goal of the CoqHoTT project is to provide a new generation of proof assistants based on this fascinating connection.

The CoqHoTT project should starts on March 2015 with a budget of 1,5M€.

#### 8.3.1.2. A4Cloud (IP)

**Participants:** Mario Südholt [coordinator], Walid Benghabrit, Ronan-Alexandre Cherrueau, Rémi Douence, Hervé Grall, Jean-Claude Royer, Mohamed Sellami.

The integrated project “Accountability for the Cloud” (A4Cloud) is coordinated by HP Labs, UK, and fosters cooperation of a consortium of five industrial and eight academic partners. It has been started in Oct. 2012 for a duration of 42 months.

A4Cloud focuses on accountability properties for the cloud and other future internet services as the most critical prerequisite for effective governance and control of corporate and private data processed by cloud-based IT services. The research being conducted in the project will increase trust in cloud computing by devising methods and tools, through which cloud stakeholders can be made accountable for the privacy and confidentiality of information held in the cloud. These methods and tools will combine risk analysis, policy enforcement, monitoring and compliance auditing. They will contribute to the governance of cloud activities, providing transparency and assisting legal, regulatory and socio-economic policy enforcement. For further information, see http://www.a4cloud.eu. ASCOLA, whose financial support consists of 550 K€, is mainly involved in the sub-projects on the enforcement of accountability and security policies, as well as tool validation efforts.

This year we have proposed new logic-based and language-level means for the formal specification and implementation of accountability properties (see 6.3).

## 8.4. International Initiatives

### *8.4.1. Inria Associate Teams*

#### *8.4.1.1. REAL*

Title: Reasoning about Aspect-oriented Programs and security In Distributed Systems

International Partner (Institution - Laboratory - Researcher):

Universidad de Chile (CHILI)

Duration: 2010-2016

See also: http://real.gforge.inria.fr

While Aspect-Oriented Programming offers promising mechanisms for enhancing the modularity of software, this increased modularity raises new challenges for systematic reasoning. This project studies means to address fundamental and practical issues in understanding distributed aspect-oriented programs by focusing on the issue of security. To this end, the project tackles three complementary lines of work: 1. Designing a core calculus to model distributed aspect-oriented programming languages and reason about programs written in these languages. 2. Studying how aspects can be used to enforce security properties in a distributed system, based upon guarantees provided by the underlying aspect infrastructure. 3. Designing and developing languages, analyses and runtime systems for distributed aspects based on the proposed calculus, therefore enabling systematic reasoning about security. These lines of work are interconnected and confluent. A concrete outcome of RAPIDS will be prototypes for two concrete distributed aspect-oriented extensions of languages increasingly used by current practitioners: Javascript and Java/Scala.

### *8.4.2. Inria International Partners*

#### *8.4.2.1. Informal International Partners*

Apart from the Inria associate team rapids with the Pleiad group (Prof. Éric Tanter) at U. Chile, the Ascola team has formalized cooperations, notably in the context of co-financed and co-supervised PhD theses with the PROG group (Prof. Wofgang de Meuter) at VU Brussel, Belgium, and the Software Technology group (Prof. Mira Mezini) at TU Darmstadt, Germany.

Furthermore, the Ascola team has long-term cooperations that resulted in common results in 2014, typically joint publications or common software artifacts, with partners from the AIST research institute (Dr. Takahiro Hirofuchi) and U. of Bogota, Colombia (Prof. Rubby Casallas).

## 8.5. International Research Visitors

### *8.5.1. Visits of International Scientists*

#### *8.5.1.1. Internships*

Gustavo Soto Ridd has done an Inria master internship advised by Nicolas Tabareau from August to November 2014. The goal of the internship was to go beyond the work on aspectual session types 6.4.

#### *8.5.1.2. Researchers*

Dr. Takahiro Hirofuchi, Researcher at AIST (Japan) spent one week in June 2014 to prepare a journal submission related to the Virtualization extensions we made in 2013 in Simgrid. The article is under review.

# 9. Dissemination

## 9.1. Promoting Scientific Activities

### *9.1.1. Scientific events organisation*

#### *9.1.1.1. General chair, scientific chair*

- A. Lebre and M. Sellami organized cloudDays@Nantes. Supported by the ASR System consortium and the Inria Large Scale Initiative Hemera, this two day national event gathered 25 researchers and phd students to discuss about latest results in Virtualization and Cloud Computing [4].
- A. Lebre was local chair of the 2014 Simgrid Users Days (30 participants).
- J.-C. Royer and M. Sellami were co-chair of the first PACS track (Privacy and Accountability for Software and Cloud Services) at WETICE 2014.

#### *9.1.1.2. Member of the organizing committee*

- Green Lab Center: T. Ledoux and J.-M. Menaud are members of the board of the Green Lab Center association. This association promotes and disseminates Green IT practices and research prototypes to the world of education, research and companies [5].
- M. Südholt has been a member of the steering committee of the international conference Modularity.

### *9.1.2. Scientific events selection*

#### *9.1.2.1. Member of the conference program committee*

- A. Lebre was member of the program committees of IEEE IC2E 2014, SCRAMBL 2014, IEEE BigData 2014, IEEE CloudCom 2014, and IEEE BDCLoud 2014.
- T. Ledoux was member of the program committees of the following conferences: the 29th Symposium On Applied Computing (SAC'14) - track Software Engineering Aspects of Green Computing, the 3rd International Conference on Eco-friendly Computing and Communication Systems (ICECCS'14), the 13th Workshop on Adaptive and Reflective Middleware (ARM'14) @ Middleware, the workshop CrossCloud'14 @ INFOCOM, the workshop CrossCloud Brokers'14 @ Middleware, the 3nd International Workshop on Green and Sustainable Software (GREENS'14) @ ICSE.
- J.-M. Menaud has served on the program committee of the third SMARTGREENS 2014, IEEE ICC SAC,The Fifth CLOUD COMPUTING 2014, the third CAGing 2014, the IEEE Globecom 2014, the 5th IEEE ICICS, the first SDS, VHPC'14 and CFSE-10.
- J. Noyé has been a member of the program committee of Modularity '14.
- J.-C. Royer was a member of the program committes of WETICE 2014, CAL 2014, CIEL 2014, ICIS 2014, and JLDP 2014. He participated in the evaluation of a project for the Israel Science Foundation.

### *9.1.3. Journal*

#### *9.1.3.1. Member of the editorial board*

- M. Südholt has been a member of the editorial board of the Springer journal "Transactions of Aspect-Oriented Software Development" (TAOSD).

#### *9.1.3.2. Reviewer*

[4] CloudDays@Nantes, 2014.
[5] Green Lab Center

- A. Lebre has been a reviewer for the JPDC and the Cluster Computing Journals.
- J. Noyé has been a reviewer for “Transactions on Aspect-Oriented Software Development”, “Software: Practice and Experience” and “Science of Computer Programming”.
- M. Südholt has been a reviewer for IEEE “Transactions in Software Engineering” (TSE) and the Springer journal “Transactions on Aspect-Oriented Software Development” (TOASD).

## 9.2. Teaching - Supervision - Juries

### *9.2.1. Teaching*

The team is involved in the following undergraduate and graduate-level programs at Mines Nantes and University of Nantes (the institutions all of eaching staff belongs to):

- The team is a main contributor to the **engineering program of EMN**.
- Within this engineering program, the team is steering, chairing and the main contributor to a two-year **graduate-level informatics specialization**. H. Grall is managing this program.
- Since 2009 our team has defined and set up a new three-year **engineering program on software engineering**. T. Ledoux is managing this program.

The team has also been involved in the following MSc programs that have been carried out with partners from French and foreign universities:

- The team participates in the **MSc program “Alma”** on software architecture and distributed systems, a joint program steered by colleagues from University of Nantes. In this context, we are responsible for a 48-hour module on advanced software composition and take part in the program’s governing board. M. Südholt is managing the participation of Mines Nantes in this program.
- Members of the team have taught different **courses at different study levels in Rennes** mainly organized by University of Rennes and the research institutes IRISA and Inria.

ASCOLA members have taught for about 210 hours on average in 2014 (hours of presence in front of students). Hereby, we have taken into account that researchers and some professors have not taught at times. In addition, another significant part of the program is taught by temporary and external staff, whose participation is managed by ASCOLA members.

In addition, J. Noyé was deputy for teaching of the Computer Department until March, 31.

### *9.2.2. Supervision*

The team has been supervising 18 PhD thesis in 2014, of which six have been co-supervised with external partners (three with foreign partners from U. Chile; TU Darmstadt, Germany; VU Brussel, Belgium), two with another Inria team (Myriads from Rennes) and one with the French TASC team from Mines Nantes.

Six PhD theses have been defended this year. Mayleen Lacouture on chemical programming for the web; Guilhem Jaber on the extension of logics for theorem provers using forcing; Diana Allam on service interoperability and securing service compositions; Guillaume Le Louët on energy management in data centers; Ismaël Figueroa on formal approaches for correct software composition; Charles Prud’homme on the structuring of constraint solvers.

Two members of the team have been preparing an HDR in 2014 for a defense in 2014.

### *9.2.3. Juries*

- J.-C. Royer was a member of the PhD committees of Cosmin Dumitrescu (Université Paris I), Hamzeih Eyal Salman (Université de Montpellier II), Diana Allam (Mines de Nantes), Ahmad Kheir (Université de Nantes and Université Libanaise), and Hamza Samih (Université de Rennes I).
- A. Lebre was a member of the PhD committee of Alexandre Lissy, “Utilisation de méthodes formelles pour garantir des propriétés de logiciels au sein d’une distribution : exemple du noyau Linux”, University of Tours, March 2014.

- J.-M. Menaud was a reviewer of the PhD of : Gueérout Tom (Dec. 5, 2014 on “Ordonnancement sous contraintes de Qualite´ de Service dans les Clouds” in Toulouse), Sébastien Schinella (Dec. 2, 2014 on “Contribution a` l’étude de l’Efficacite´ énergétique des Services TIC” in Paris), Aurélien Wailly (Sep. 15, 2014 on “Architecture de se´curite´ de bout-en-bout pour les environnements cloud” in Paris), Djawida Dib (Jul. 7, 2014 on “Optimizing PaaS Provider Profit under Service Level Agreement Constraints” in Rennes), Tran Giang Son (Jun. 4, 2014 on “Cooperative Resource Management in the Cloud” in Grenoble), Amit Sangroya (Apr. 24, 2014 on “Towards Dependability and Performance Benchmarking for Cloud Computing Services” in Grenoble).
- M. Südholt has served as reporting member of the examination committee of Mohamed Aly at TU Darmstadt, Germany, in Nov. 2014.

## 10. Bibliography

### Major publications by the team in recent years

[1] B. De Fraine, E. Ernst, M. Südholt. *Essential AOP: The A Calculus*, in "ACM Transactions on Programming Languages and Systems (TOPLAS)", December 2012, http://hal.inria.fr/hal-00676082

[2] I. Figueroa, T. Schrijvers, N. Tabareau, É. Tanter. *Compositional Reasoning About Aspect Interference*, in "Modularity’14: 13th International Conference on Modularity", Lugano, Switzerland, April 2014, https://hal.inria.fr/hal-00919935

[3] G. Jaber, N. Tabareau, M. Sozeau. *Extending Type Theory with Forcing*, in "LICS’12 : Logic In Computer Science", Dubrovnik, Croatia, June 2012, http://hal.inria.fr/hal-00685150

[4] Y. Kouki, F. Alvares De Oliveira Jr., S. Dupont, T. Ledoux. *A Language Support for Cloud Elasticity Management*, in "CCGrid’14: IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing", Chicago, United States, May 2014, pp. 1-8, https://hal.archives-ouvertes.fr/hal-00941945

[5] J. Pastor, M. Bertier, F. Desprez, A. Lebre, F. Quesnel, C. Tedeschi. *Locality-aware Cooperation for VM Scheduling in Distributed Clouds*, in "Euro-Par’14", Porto, Portugal, August 2014, https://hal.inria.fr/hal-00991530

[6] F. Quesnel, A. Lebre, M. Südholt. *Cooperative and Reactive Scheduling in Large-Scale Virtualized Platforms with DVMS*, in "Concurrency and Computation: Practice and Experience", December 2012, http://hal.inria.fr/hal-00675315

[7] M. Sozeau, N. Tabareau. *Universe Polymorphism in Coq*, in "ITP’14: Interactive Theorem Proving", Vienna, Austria, July 2014, https://hal.inria.fr/hal-00974721

[8] N. Tabareau, M. Südholt, É. Tanter. *Aspectual Session Types*, in "Modularity’14 - 13th International Conference on Modularity", Lugano, Switzerland, April 2014, https://hal.inria.fr/hal-00872791

[9] R. Toledo, A. Núñez, É. Tanter, J. Noyé. *Aspectizing Java Access Control*, in "IEEE Transactions on Software Engineering", January 2011, http://hal.inria.fr/inria-00567489/en

[10] J. Van Ham, G. Salvaneschi, M. Mezini, J. Noyé. *JEScala: Modular Coordination with Declarative Events and Joins*, in "Modularity’14 - 13th International Conference on Modularity", Lugano, Switzerland, E. Ernst (editor), April 2014, https://hal.inria.fr/hal-00862332

## Publications of the year

### Doctoral Dissertations and Habilitation Theses

[11] D. ALLAM. *Loose coupling and substitution principle in objet-oriented frameworks for web services*, Ecole des Mines de Nantes, July 2014, https://tel.archives-ouvertes.fr/tel-01083286

[12] I. FIGUEROA. *Effective aspects : A typed monadic model to control and reason about aspect interference*, Ecole des Mines de Nantes ; Universidad de Chile. Facultad de ciencias físicas y matemáticas, April 2014, https://tel.archives-ouvertes.fr/tel-01067730

[13] M. LACOUTURE. *A Chemical Programming Language for Orchestrating Services*, Mines de Nantes, October 2014, https://tel.archives-ouvertes.fr/tel-01109582

[14] G. LE LOUËT. *Power management in virtualized data centers : Form a load scenario to the optimization of the tasks placement*, Ecole des Mines de Nantes, May 2014, https://tel.archives-ouvertes.fr/tel-01044650

[15] C. PRUD'HOMME. *Controlling propagation and search within a constraint solver*, Ecole des Mines de Nantes, February 2014, https://tel.archives-ouvertes.fr/tel-01060921

### Articles in International Peer-Reviewed Journals

[16] I. FIGUEROA, N. TABAREAU, É. TANTER. *Effective Aspects: A Typed Monadic Embedding of Pointcuts and Advice*, in "Transactions on Aspect-Oriented Software Development", 2014, https://hal.inria.fr/hal-00872782

[17] M. SELLAMI, P. DE VETTOR, M. MRISSA, D. BENSLIMANE, B. DEFUDE. *DMaaS : Syntactic, Structural and Semantic Mediation for Service Composition*, in "International Journal of Autonomous and Adaptive Communications Systems", 2014, https://hal.inria.fr/hal-00937178

[18] É. TANTER, I. FIGUEROA, N. TABAREAU. *Execution Levels for Aspect-Oriented Programming: Design, Semantics, Implementations and Applications*, in "Science of Computer Programming", February 2014, vol. 80, n$^{o}$ 1, pp. 311-342 [*DOI :* 10.1016/J.SCICO.2013.09.002], https://hal.inria.fr/hal-00872786

### International Conferences with Proceedings

[19] W. BENGHABRIT, H. GRALL, J.-C. ROYER, M. SELLAMI, M. AZRAOUI, K. ELKHIYAOUI, M. ÖNEN, A. SANTANA DE OLIVEIRA, K. BERNSMED. *A Cloud Accountability Policy Representation Framework*, in "CLOSER - 4th International Conference on Cloud Computing and Services Science", Barcelone, Spain, April 2014, https://hal.inria.fr/hal-00941872

[20] W. BENGHABRIT, H. GRALL, J.-C. ROYER, M. SELLAMI, K. BERNSMED, A. SANTANA DE OLIVEIRA. *Abstract Accountability Language*, in "IFIPTM - 8th IFIP WG 11.11 International Conference on Trust Management", Singapore, Trust Management VIII - 8th IFIP WG 11.11 International Conference, July 2014, vol. 430, pp. 229–236, https://hal.inria.fr/hal-00973399

[21] W. BENGHABRIT, H. GRALL, J.-C. ROYER, M. SELLAMI. *Accountability for Abstract Component Design*, in "EUROMICRO DSD/SEAA 2014", Verona, Italy, August 2014, https://hal.inria.fr/hal-00987165

[22] R.-A. CHERRUEAU, M. SÜDHOLT. *Enforcing Expressive Accountability Policies*, in "WETICE - IEEE International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises", Parma, Italy, Sumitra Reddy, June 2014, pp. 333–338 [*DOI :* 10.1109/WETICE.2014.71], https://hal.inria.fr/hal-00967398

[23] R. DOUENCE, N. TABAREAU. *Lazier Imperative Programming*, in "Principles and Practice of Declarative Programming (PPDP)", Canterbury, United Kingdom, September 2014, https://hal.inria.fr/hal-01016565

[24] I. FIGUEROA, T. SCHRIJVERS, N. TABAREAU, É. TANTER. *Compositional Reasoning About Aspect Interference*, in "13th International Conference on Modularity (Modularity'14)", Lugano, Switzerland, April 2014, https://hal.inria.fr/hal-00919935

[25] I. FIGUEROA, N. TABAREAU, É. TANTER. *Effect Capabilities For Haskell*, in "Brazilian Symposium on Programming Languages (SBLP)", Maceio, Brazil, September 2014, https://hal.inria.fr/hal-01038053

[26] M. S. HASAN, Y. KOUKI, T. LEDOUX, J.-L. PAZAT. *Cloud Energy Broker: Towards SLA-driven Green Energy Planning for IaaS Providers*, in "HPCC - IEEE Internatonal Conference on High Performance Computing and Communications", France, August 2014, pp. 1-8, https://hal.archives-ouvertes.fr/hal-01015811

[27] Y. KOUKI, F. ALVARES DE OLIVEIRA JR., S. DUPONT, T. LEDOUX. *A Language Support for Cloud Elasticity Management*, in "CCGrid - IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing", Chicago, United States, May 2014, pp. 1-8, https://hal.archives-ouvertes.fr/hal-00941945

[28] A. LEBRE, G. BERVIAN BRAND. *GBFS: Efficient Data-Sharing on Hybrid Platforms. Towards adding WAN-Wide elasticity to DFSes.* , in "WPBA Workshop in Proceedings of 26th International Symposium on Computer Architecture and High Performance Computing", Paris, France, IEEE, October 2014, https://hal.inria.fr/hal-01085233

[29] J. PASTOR, M. BERTIER, F. DESPREZ, A. LÈBRE, F. QUESNEL, C. TEDESCHI. *Locality-aware Cooperation for VM Scheduling in Distributed Clouds*, in "Euro-Par 2014", Porto, Portugal, August 2014, https://hal.inria.fr/hal-00991530

[30] M. SELLAMI, J.-C. ROYER, W. BENGHABRIT. *Accountability for Data Protection*, in "International Workshop on Computational Intelligence for Multimedia Understanding", Paris, France, November 2014, https://hal.archives-ouvertes.fr/hal-01084890

[31] M. SOZEAU, N. TABAREAU. *Universe Polymorphism in Coq*, in "Interactive Theorem Proving", Vienna, Austria, July 2014, https://hal.inria.fr/hal-00974721

[32] N. TABAREAU, M. SÜDHOLT, É. TANTER. *Aspectual Session Types*, in "Modularity - 13th International Conference on Modularity", Lugano, Switzerland, April 2014 [*DOI :* 10.1145/2577080.2577085], https://hal.inria.fr/hal-00872791

[33] J. VAN HAM, G. SALVANESCHI, M. MEZINI, J. NOYÉ. *JEScala: Modular Coordination with Declarative Events and Joins*, in "Modularity '14 - 13th International Conference on Modularity", Lugano, Switzerland, E. ERNST (editor), April 2014, https://hal.inria.fr/hal-00862332

### Scientific Books (or Scientific Book chapters)

[34] H. ARBOLEDA, R. CASALLAS, J. CHAVARRIAGA, J.-C. ROYER. *Software Architecture for Product Lines*, in "Software Architecture 1", M. OUSSALAH (editor), Wiley-ISTE, April 2014, pp. 171-210, https://hal.archives-ouvertes.fr/hal-00997673

[35] A. LÈBRE, J. PASTOR, M. BERTIER, F. DESPREZ, J. ROUZAUD-CORNABAS, C. TEDESCHI, A.-C. ORGERIE, F. QUESNEL, G. FEDAK. *Beyond The Clouds, How Should Next Generation Utility Computing Infrastructures Be Designed?*, in "Cloud Computing: Challenges, Limitations and R&D Solutions", Z. MAHMOOD (editor), Springer, November 2014, https://hal.inria.fr/hal-01067888

### Research Reports

[36] R. DOUENCE, N. TABAREAU. *Lazier Imperative Programming*, July 2014, n$^{o}$ RR-8569, https://hal.inria.fr/hal-01025633

### Other Publications

[37] J. FABRY, T. DINKELAKER, J. NOYÉ, É. TANTER. *A Taxonomy of Domain-Specific Aspect Languages*, October 2014, Accepted for publication in ACM Computing Surveys, https://hal.inria.fr/hal-01085063

## References in notes

[38] M. AKŞIT, S. CLARKE, T. ELRAD, R. E. FILMAN (editors). *Aspect-Oriented Software Development*, Addison-Wesley Professional, September 2004

[39] C. ALLAN, P. AVGUSTINOV, A. S. CHRISTENSEN, L. HENDREN, S. KUZINS, O. LHOTÁK, O. DE MOOR, D. SERENI, G. SITTAMPALAM, J. TIBBLE. *Adding trace matching with free variables to AspectJ*, in "ACM Conference on Object-Oriented Programming, Systems and Languages (OOPSLA)", R. P. GABRIEL (editor), ACM Press, 2005

[40] R. ALLEN, D. GARLAN. *A Formal Basis for Architectural Connection*, in "ACM Transactions on Software Engineering and Methodology", July 1997, vol. 6, n$^{o}$ 3, pp. 213–49

[41] J. H. ANDREWS. *Process-Algebraic Foundations of Aspect-Oriented Programming*, in "Proceedings of the 3rd International Conference on Metalevel Architectures and Separation of Crosscutting Concerns", Lecture Notes in Computer Science, 2001, vol. 2192, pp. 187–209

[42] T. H. AUSTIN, C. FLANAGAN. *Multiple facets for dynamic information flow*, in "Proceedings of the 39th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages", New York, USA, POPL '12, ACM, 2012, pp. 165–178, http://doi.acm.org/10.1145/2103656.2103677

[43] L. D. BENAVIDES NAVARRO, M. SÜDHOLT, W. VANDERPERREN, B. DE FRAINE, D. SUVÉE. *Explicitly distributed AOP using AWED*, in "Aspect-Oriented Software Development (AOSD)", ACM Press, March 2006, pp. 51-62

[44] G. S. BLAIR, G. COULSON, P. ROBIN, M. PAPATHOMAS. *An architecture for next generation middleware*, in "Proceedings of the IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing", Springer-Verlag, 1998

[45] A. BRACCIALIA, A. BROGI, C. CANAL. *A formal approach to component adaptation*, in "Journal of Systems and Software", 2005

[46] S. Capecchi, I. Castellani, M. Dezani-Ciancaglini, T. Rezk. *Session Types for Access and Information Flow Control*, in "CONCUR 2010 - Concurrency Theory, 21th International Conference, CONCUR 2010, Paris, France, August 31-September 3, 2010. Proceedings", P. Gastin, F. Laroussinie (editors), Lecture Notes in Computer Science, Springer, 2010, vol. 6269, pp. 237–252, http://dx.doi.org/10.1007/978-3-642-15375-4_17

[47] E. M. Clarke, O. Grumberg, D. A. Peled. *Model Checking*, The MIT PressCambridge, Massachusetts, 1999

[48] A. Colyer, A. Clement. *Large-scale AOSD for Middleware*, in "Proceedings of the 3rd ACM Int. Conf. on Aspect-Oriented Software Development (AOSD), Lancaster", K. Lieberherr (editor), ACM Press, 2004, pp. 56–65

[49] F. DeRemer, H. H. Kron. *Programming-in-the-large versus programming-in-the-small*, in "IEEE Transactions on Software Engineering", 1976, vol. SE-2, n$^{o}$ 2, pp. 80-86

[50] G. Decker, O. Kopp, F. Leymann, M. Weske. *BPEL4Chor: Extending BPEL for Modeling Choreographies*, in "IEEE International Conference on Web Services (ICWS 2007)", IEEE Computer Society, 2007, pp. 296–303

[51] E. W. Dijkstra. *On the role of scientific thought*, in "Selected Writings on Computing: A Personal Perspective", Springer-Verlag, 1974, pp. 60–66, Published in 1982

[52] R. Douence, P. Fradet, M. Südholt. *A framework for the detection and resolution of aspect interactions*, in "Proceedings of the ACM SIGPLAN/SIGSOFT Conference on Generative Programming and Component Engineering (GPCE'02)", Lecture Notes in Computer Science, Springer-Verlag, October 2002, vol. 2487, pp. 173–188, http://hal.inria.fr/inria-00072153

[53] R. Douence, P. Fradet, M. Südholt. *Trace-Based Aspects*, in "Aspect-Oriented Software Development", M. Akşit, S. Clarke, T. Elrad, R. E. Filman (editors), Addison-Wesley, 2004, pp. 201-218

[54] R. Douence, O. Motelet, M. Südholt. *A formal definition of crosscuts*, in "Proceedings of the 3rd International Conference on Metalevel Architectures and Separation of Crosscutting Concerns", Lecture Notes in Computer Science, Springer-Verlag, 2001, vol. 2192, pp. 170–186

[55] R. Douence, D. Le Botlan, J. Noyé, M. Südholt. *Concurrent Aspects*, in "Proc. of the Int. ACM Conf. on Generative Programming and Component Engineering (GPCE)", ACM Press, October 2006, pp. 79-88

[56] H. Foster, S. Uchitel, J. Magee, J. Kramer. *Model-based Verification of Web Service Compositions*, in "Proceedings of the 18th IEEE Int. Conf. on Automated Software Engineering (ASE'03)", IEEE Computer Society, 2003, pp. 152–163

[57] A. Fuggetta, G. P. Picco, G. Vigna. *Understanding Code Mobility*, in "IEEE Transactions on Software Engineering", May 1998, vol. 24, n$^{o}$ 5, pp. 342–361

[58] K. Honda, N. Yoshida, M. Carbone. *Multiparty asynchronous session types*, in "Proceedings of the 35th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2008, San Francisco, California, USA, January 7-12, 2008", G. C. Necula, P. Wadler (editors), ACM, 2008, pp. 273–284, http://www.doc.ic.ac.uk/~yoshida/multiparty/multiparty.pdf

[59] G. KICZALES, E. HILSDALE, J. HUGUNIN, M. KERSTEN, J. PALM, W. G. GRISWOLD. *An Overview of AspectJ*, in "ECOOP 2001 — Object-Oriented Programming 15th European Conference, Budapest Hungary", Berlin, J. L. KNUDSEN (editor), Lecture Notes in Computer Science, Springer-Verlag, June 2001, vol. 2072, pp. 327–353, http://www.eclipse.org/aspectj/

[60] G. KICZALES. *Aspect Oriented Programming*, in "Proc. of the Int. Workshop on Composability Issues in Object-Orientation (CIOO'96) at ECOOP", July 1996, Selected paper published by dpunkt press, Heidelberg, Germany

[61] G. KICZALES, J. DES RIVIERES, DANIEL G. BOBROW. *The Art of the Meta-Object Protocol*, MIT PressCambridge (MA), USA, 1991

[62] J. KIENZLE, R. GUERRAOUI. *AOP - Does It Make Sense? The Case of Concurrency and Failures*, in "16th European Conference on Object-Oriented Programming (ECOOP'2002)", Malaga, Spain, B. MAGNUSSON (editor), Lecture Notes in Computer Science, Springer-Verlag, 2002

[63] T. LEDOUX. *OpenCorba: a Reflective Open Broker*, in "ACM Meta-Level Architectures and Reflection, Second International Conference, Reflection'99", Saint-Malo, France, P. COINTE (editor), Lecture Notes in Computer Science, Springer-Verlag, July 1999, vol. 1616, pp. 197–214

[64] M. MCILROY. *Mass produced software components*, in "Mass produced software components", Garmish, Germany, P. NAUR, B. RANDELL (editors), NATO Science Committee, October 1968, pp. 138-155

[65] N. MEDVIDOVIC, R. N. TAYLOR. *A Classification and Comparison Framework for Software Architecture Description Languages*, in "IEEE Transactions on Software Engineering", January 2000, vol. 26, n$^{o}$ 1, pp. 70-93

[66] M. MERNIK, J. HEERING, A. M. SLOANE. *When and How to Develop Domain-Specific Languages*, in "ACM Computing Surveys", December 2005, vol. 37, n$^{o}$ 4, pp. 316-344

[67] L. MIKHAJLOV, E. SEKERINSKI. *A study of the fragile base class*, in "A study of the fragile base class", Brussels, Belgium, E. JUL (editor), Lecture Notes in Computer Science, July 1998, vol. 1445, pp. 355-382

[68] D. H. NGUYEN, M. SÜDHOLT. *VPA-based aspects: better support for AOP over protocols*, in "4th IEEE International Conference on Software Engineering and Formal Methods (SEFM'06)", IEEE Computer Society Press, September 2006

[69] O. NIERSTRASZ. *Regular Types for Active Objects*, in "Object-Oriented Software Composition", O. NIERSTRASZ, D. TSICHRITZIS (editors), Prentice Hall, 1995, chap. 4, pp. 99–121

[70] M. NISHIZAWA, S. CHIBA, M. TATSUBORI. *Remote Pointcut - A Language Construct for Distributed AOP*, in "Proceedings of the 3rd ACM Int. Conf. on Aspect-Oriented Software Development (AOSD), Lancaster", ACM Press, 2004

[71] D. L. PARNAS. *On the criteria for decomposing systems into modules*, in "Communications of the ACM", December 1972, vol. 15, n$^{o}$ 12, pp. 1053-1058

[72] S. PEARSON. *Toward Accountability in the Cloud*, in "Internet Computing, IEEE", July-Aug. 2011, vol. 15, n° 4, pp. 64-69, http://dx.doi.org/10.1109/MIC.2011.98

[73] F. PLASIL, S. VISNOVSKY. *Behavior Protocols for Software Components*, in "Transactions on Software Engineering", January 2002, vol. 28, n° 9

[74] F. PUNTIGAM. *Coordination Requirements Expressed in Types for Active Objects*, in "ECOOP'97—Object-Oriented Programming", M. AKŞIT, S. MATSUOKA (editors), Lecture Notes in Computer Science, Springer-Verlag, 1997, vol. 1241, pp. 367–388

[75] M. SHAW, D. GARLAN. *Software Architecture: Perspectives on an Emerging Discipline*, Prentice-Hall, 1996

[76] B. C. SMITH. *Reflection and Semantics in LISP*, Xerox Palto Alto Research CenterPalo Alto, 1984, n° P84-00030

[77] S. SOARES, E. LAUREANO, P. BORBA. *Implementing distribution and persistence aspects with AspectJ* , in "Proceedings of the 17th ACM conference on Object-oriented programming, systems, languages, and applications (OOPSLA-02)", C. NORRIS, J. J. B. FENWICK (editors), ACM SIGPLAN Notices, ACM Press, November 4–8 2002, vol. 37, 11, pp. 174–190

[78] S. SUNDARESWARAN. *Ensuring Distributed Accountability for Data Sharing in the Cloud*, in "Dependable and Secure Computing", 2012, vol. 9, http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6165313

[79] R. J. WALKER, K. VIGGERS. *Implementing Protocols via Declarative Event Patterns*, in "Proceedings of the ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE-12)", ACM Press, 2004, pp. 159 - 169

[80] M. WAND, G. KICZALES, C. DUTCHYN. *A Semantics for Advice and Dynamic Join Points in Aspect-Oriented Programming*, in "ACM Transactions on Programming Languages and Systems (TOPLAS)", 2004, vol. 26, n° 5, pp. 890–910

[81] D. M. YELLIN, R. E. STROM. *Protocol specifications and component adaptors*, in "ACM Transactions of Programming Languages and Systems", March 1997, vol. 19, n° 2, pp. 292–333

[82] A. VAN DEURSEN, P. KLINT, J. VISSER. *Domain-Specific Languages: An Annotated Bibliography*, in "ACM SIGPLAN Notices", June 2000, vol. 35, n° 6, pp. 26-36