IN PARTNERSHIP WITH:
CNRS

Ecole des Mines de Nantes

Activity Report 2012

Project-Team ASCOLA

Aspect and composition languages

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

RESEARCH CENTER
Rennes - Bretagne-Atlantique

THEME
Distributed Systems and Services

Table of contents

1. Members 1
2. Overall Objectives 2
2.1. Presentation 2
2.2. Highlights of the Year 2
3. Scientific Foundations 2
3.1. Overview 2
3.2. Software Components 3
3.3. Aspect-Oriented Programming 3
3.4. Protocols 4
3.5. Patterns 4
3.6. Domain-Specific Languages 5
3.7. Distribution and Concurrency 5
4. Application Domains 6
4.1. Enterprise Information Systems and Services 6
4.2. Cluster, Grid and Cloud Computing 7
4.3. Pervasive Systems 7
5. Software 7
5.1. AWED 7
5.2. btrCloud (and Entropy) 7
5.3. ECaesarJ, EJava and EScala 8
5.4. FPath and FScript 8
5.5. WildCAT 9
6. New Results 9
6.1. Software composition 9
6.1.1. Program transformation and formal properties 9
6.1.1.1. Forcing in the Calculus of Constructions and Coq 10
6.1.1.2. Invertible transformations for program decompositions 10
6.1.2. Service-oriented computing 10
6.1.2.1. Uniform service model 10
6.1.2.2. A type system for services 10
6.1.2.3. Criojo: a pivot language for services 11
6.1.3. Languages and composition models 11
6.1.3.1. Software product lines and model composition 11
6.1.3.2. Expressive language support for numerical constraint based programming 11
6.1.4. Analysis and test of C programs 11
6.2. Aspect-Oriented Programming 11
6.2.1. Formal models for AOP 12
6.2.1.1. The A Calculus 12
6.2.1.2. A category-theoretic foundation of aspects 12
6.2.2. Programming languages for aspects and related paradigms 12
6.2.2.1. Concurrent multi-paradigm programming with EScala 12
6.2.2.2. Monascheme: modular prototyping of aspect languages 12
6.2.2.3. Structuring computations with aspect-based membranes 13
6.3. Cloud applications and infrastructures 13
6.3.1. SLA Management for Cloud elasticity 13
6.3.2. Fully distributed and autonomous virtualized environments 13
6.3.3. Energy-efficient Cloud applications and infrastructures 13
7. Bilateral Contracts and Grants with Industry 14
8. Partnerships and Cooperations 14

8.1. National Initiatives 14
8.1.1. ANR 14
8.1.1.1. CESSA: Compositional Evolution of Secure Services with Aspects (ANR/ARPEGE) 14
8.1.1.2. Entropy (ANR/Emergence) 14
8.1.1.3. MyCloud (ANR/ARPEGE) 15
8.1.1.4. SONGS (ANR/INFRA) 15
8.1.2. FUI 15
8.1.3. FSN 16
8.2. European Initiatives 16
8.2.1. FP7 Projects 16
8.2.2. Collaborations in European Programs, except FP7 16
8.2.2.1. SCALUS: SCALing by means of Ubiquitous Storage (MC ITN) 16
8.2.2.2. COST IC0804 17
8.3. International Initiatives 17
8.4. International Research Visitors 18
9. Dissemination **18**
9.1. Scientific Animation 18
9.1.1. Event organization, animation of scientific community 18
9.1.1.1. International 18
9.1.1.2. National 18
9.1.2. Committee participations, reviewing activities, collective duties, etc. 18
9.2. Teaching - Supervision - Juries 19
9.2.1. Teaching 20
9.2.2. Supervision 20
9.2.3. Juries 20
10. Bibliography **20**

Project-Team ASCOLA

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

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

1. Members

Research Scientist

Nicolas Tabareau [Junior Researcher Inria]

Faculty Members

Tony Bourdier [Associate Professor (temporary), until Aug. 12]
Pierre Cointe [Professor, EMN, HdR]
Rémi Douence [Associate Professor, Researcher Inria on leave from EMN until Aug. 12]
Hervé Grall [Associate Professor, EMN]
Adrien Lèbre [Associate Professor, EMN]
Thomas Ledoux [Associate Professor, EMN]
Jean-Marc Menaud [Associate Professor, EMN, HdR]
Jacques Noyé [Vice Team Leader: Associate Professor, EMN]
Jean-Claude Royer [Professor, EMN, HdR]
Remi Sharrock [Associate Professor (temporary), until Aug. 12]
Mario Südholt [Team Leader: Professor, EMN, HdR]

Engineers

Thierry Bernard [EMN (ANR Emergence Entropy) grant]
Omar Chebaro [Inria]
Clotilde Massot [Inria (ADT Vasp)]

PhD Students

Akram Ajouli [EMN grant, since Oct. 10]
Diana Allam [ANR Cessa grant, since Oct. 10]
Frederico Alvares [MESR grant, since Oct. 09]
Gustavo Bervian Brand [EU MCITN Scalus grant, since Sep. 10]
Ronan-Alexandre Cherrueau [EU IP A4Cloud & PdL region grant, since Oct. 12]
Simon Dupont [Cifre Sigma grant, since Nov. 12]
Ismael Figueroa [CONICYT grant, since Jan. 12]
Guilhem Jaber [ENS grant, since Sep. 10]
Yousri Kouki [ANR Mycloud grant, since Dec. 10]
Mayleen Lacouture [IST Ample grant, since Oct. 08]
Guillaume Le Louët [EMN grant, since Oct. 10]
Florent Marchand de Kerchove [CominLabs grant, since Oct. 12]
Ismael Mejía [EMN grant, since Jan. 09]
Syed Asad Ali Navqi [EMN & Univ. of Lancaster grant, since Nov. 08]
Jonathan Pastor [EMN grant, since Oct. 12]
Rémy Pottier [ANR SelfXL grant, until Sep. 12]
Flavien Quesnel [EMN grant, since Oct. 09]
Jurgen Van Ham [Armines & TUD grant, since Sep. 10]

Administrative Assistants

Diana Gaudin [Part-time (33%), until Oct. 12]
Florence Rogues [Part-time (33%), from Nov. 12]
Cécile Derouet [Part-time (33%)]

2. Overall Objectives

2.1. Presentation

The ASCOLA project-team aims at harnessing and developing advanced application structuring mechanisms, and supporting the transformational development of correct large-scale applications as well as their valid evolution throughout their entire life cycle. We apply a language-based approach to achieve this goal, defining new languages in order to represent architectural as well as implementation level abstractions and exploit formal methods to ensure correctness.

Concretely, we investigate expressive aspect languages to modularize crosscutting concerns. Those languages enable sophisticated relationships between execution events to be formulated and manipulated directly at the language level. We study how to reconcile invasive accesses by aspects with strongly encapsulated software entities. Furthermore, we foster the transformational development of implementations from higher-level architectural software representations using domain-specific languages. Finally, we focus on abstractions and development methods for distributed and concurrent systems, in particular flexible and energy-efficient infrastructures.

Our results are subjected to validation in the context of four main application domains: enterprise information systems, service-oriented architectures, cluster/grid, cloud applications, and pervasive systems.

2.2. Highlights of the Year

This year we have produced three major scientific results. Concerning the foundations of programming, we have extending the Calculus of Construction, the type theory underlying the Coq theorem prover by new logical principles that haven't been tractable before [22], see Sec. 6.1. We have also extended the A Calculus, the currently most comprehensive foundational calculus for AspectJ-like and history-based aspect models, by a non-deterministic semantics and have provided a type soundness proof that is close to standard mathematical reasoning and automated using Coq [12], see Sec. 6.2. In the domain of efficient Cloud infrastructures, we have proposed DVMS, a fully-distributed and autonomous system for VM scheduling, that includes one of the currently most highly-scalable scheduling algorithm [13], see Sec. 6.3.

ASCOLA is part of the new EU FP7 IP project A4Cloud on Accountability for the Cloud, a project with 13 industrial and academic partners, see Sec. 8.2.

Finally, ASCOLA members have organized the Grid'5000 international winter school at École de Mines de Nantes, which brought together seventy, mostly European, PhD students and senior researchers, see Sec. 9.1.

3. Scientific Foundations

3.1. Overview

Since we mainly work on new software structuring concepts and programming language design, we first briefly introduce some basic notions and problems of software components (understood in a broad sense, i.e., 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 that are relevant to our work.

3.2. Software Components

Modules and services. The idea that building *software components*, i.e., composable prefabricated and parametrized software parts, was key to create an effective software industry was realized very early [67]. 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 [76], [48]. 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 a first step, it is possible to reason locally, about the consistency between the module implementation and the module interface. In a second step, it is possible to reason about composing modules by only considering their interfaces. Modern module systems also consider types as module elements and consider, typically static, modules as a unit of separate compilation, with the most sophisticated ones also supporting modules parametrized by modules [65].

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 [71].

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* [79] of a software system describes the system as a composition of *components* and *connectors*, where the connectors capture the *interaction protocols* between the components [39]. 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 [68]. 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.

3.3. Aspect-Oriented Programming

The main driving force for the structuring means, such as components and modules, is the quest for clean *separation of concerns* [50] 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 [80], [62] 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.

Aspect-Oriented Software Development [61], [37] 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 [60], 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 [81]. 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 [47]. 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 [83]) and provide only limited support for the analysis of fundamental aspect properties.

Recently, first approaches have been put forward to tackle these problems, in particular, in the context of so-called *stateful* or *history-based aspect languages* [51], [52], 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 [75], [41], [54], provide more concise formal semantics for aspects and enable analysis of their properties [40], [53], [51], [38]. Due to the novelty of these approaches, they represent, however, only first results and many important questions concerning these fundamental issues remain open.

3.4. Protocols

Today, protocols constitute a frequently used means to precisely define, implement, and analyze contracts 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 [74], component interactions [84], [77] and service orchestrations [49] 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 [44], [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 [59], or context-free or turing-complete expressiveness [78], [43]. 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.

Furthermore, first approaches for the definition of *aspects over protocols* have been proposed, as well as over regular structures [51] and non-regular ones [82], [73]. The modification of interaction protocols by aspects seems highly promising for the *integration of aspects and components*.

3.5. Patterns

Patterns provide a kind of abstraction that is complementary to the modularization mechanisms discussed above. They have been used, in particular, to define general *architectural styles* either by defining entire computation and communication topologies [72], connectors between (complex) software artifacts [69], or (based on, possibly concretizations of, *design patterns* [58]) as building blocks for object-oriented software architectures. The resulting pattern-based architectures are similar to common component-based architectures and are frequently used to implement the latter, see, for instance, Sun's J2EE patterns.

Patterns have also been used to implement architectural abstractions. This is the case, for instance, for the numerous variants of the *publish/subscribe pattern* [55] as well as the large set of so-called *skeletons* [46], that is, patterns for the implementation of distributed and concurrent systems. While these patterns are essentially similar to architecture-level patterns, their fine-grained application to multiple code entities often results in crosscutting code structures. An important open issue consists in the lack of pattern-based representations for the implementation of general distributed applications — in sharp contrast to their use for the derivation of massively parallel programs.

3.6. Domain-Specific Languages

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 [70], [85].

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.

3.7. 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* [42], [64]. 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 [63] and the modularization of the distribution concern itself [81]. 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 [40], [54].

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 2012, we have developed techniques for the Service-Level Agreement (SLA) management for Cloud elasticy, see Sec. 6.3, as well as models and type systems for service-oriented systems, see Sec. 6.1. Furthermore, we take part in a starting new European project A4Cloud on accountability challenges, that is, the responsible stewardship of third-party data and computations, see Sec. 8.2.

4.2. Cluster, Grid and Cloud Computing

Cluster, Grid and more recently Cloud computing platforms aim at delivering a larger capacity of computing power compared to a single computer configuration. This capacity can be used to improve performance (for scientific applications) or availability (e.g., for Internet services hosted by a data center). These distributed infrastructures consist of a group of coupled computers that work together. This group can 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 calls for automation of the adaptation process. 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 2012, we have developed the DVMS system, which contains one of the most highly scalable scheduling algorithm for virtual machines; we have also generated several results on the energy efficient management of Cloud applications and infrastructures, see Sec. 6.3.

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 2012, we have developed the language EScala, which integrates reactive programming through events with aspect-oriented and object-oriented mechanisms, see Sec. 6.1.

5. Software

5.1. AWED

Participants: Mario Südholt [correspondent], Ismael Mejia.

Aspect-oriented programming, distributed programming, event-based programming, invasive patterns

The model of Aspects With Explicit Distribution (AWED) supports the modularization of crosscutting functionalities of distributed applications. It addresses the problem that common aspect systems do not provide features for distributed programming. It notably features three main aspect abstractions: remote pointcuts, remotely-executed advice, and distributed aspects.

The AWED system has also been employed in the CESSA project proposal (see Sec. 8.1) as a basis for our work on the secure evolution of service-oriented architectures.

AWED is available at http://awed.gforge.inria.fr.

5.2. btrCloud (and Entropy)

Participants: Jean-Marc Menaud [correspondent], Rémy Pottier, Clotilde Massot, Guillaume Le Louët, Thierry Bernard, 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 center. 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 or on the long run and predict future trends. This feature includes a performances, an analysis and a trends board.

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, of course depending on the context. The latter enhances provides optimal quality of service for the applications that are hosted in the virtualized data-center.

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

5.3. ECaesarJ, EJava and EScala

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

Symmetric AOP, features, software product lines, inheritance, virtual classes, propagating mixin composition, event-based programming, events, declarative events, state machines, CaesarJ, Java, Scala

ECaesarJ is a language developed in the context of the European project AMPLE, as joint work with the *Technische Universität Darmstadt* (TUD). The basic objective was to provide support for directly mapping the high-level features defined by a software product line onto implementation-level features, beyond standard feature-oriented programming. But the language has much wider applications. ECaesarJ can actually be seen as a language which smoothly integrates Object-Oriented Programming, Feature-Oriented Programming, Aspect-Oriented Programming, and Event-based Programming.

It is an extension of Java with *virtual classes* and *propagating mixin composition* (as its ancestor CaesarJ, developed at TUD), but also *declarative events* and *state machines*. Unlike AspectJ, ECaesarJ does not include a class-like concept of aspect. Instead, it deals with pointcuts and pieces of advice as (implicit) events and event handlers, which are standard class members. This makes it possible to use standard inheritance to reuse and refine them. Explicit events can also be used when events must be explicitly triggered as in traditional event-based programming. Finally, in the same way as pointcuts can be composed using logical operators, *declarative events* can be defined as a composition of other events.

This provides a symmetric version of AOP where virtual classes can be used to deal with structural aspects whereas events can be used to deal with behavioral aspects.

In ECaesarJ, a class can also include, as class members, state transitions. Combining this with virtual classes makes it possible to define, at the programming language level, refinable hierarchical state machines. The combination of state machines and events provides, in particular, effective language support for the State design pattern as well as a form of Event-based AOP.

EJava and EScala are more recent developments of the same ideas applied to Java and Scala, respectively. EJava benefits from Java tooling with an eclipse plugin developed with the Spoofax Language Workbench. Unlike EJava and ECaesarJ, EScala makes it possible to dynamically register and unregister event handlers. It also benefits from a more efficient, compiler-based, implementation. As ECaesarJ, EScala is joint work with TUD.

Prototype implementations of these languages are available through http://ecaesarj.gforge.inria.fr/.

5.4. FPath and FScript

Participants: Thomas Ledoux [correspondent], Frederico Alvares.

dynamic reliable reconfiguration, self-adaptive components, Fractal, autonomic computing

FPath and FScript are two domain-specific languages (DSLs) dealing respectively with the navigation and the dynamic reconfiguration of Fractal architectures. *FPath* is a DSL for querying Fractal architectures. It is restricted to the introspection of architectures by browsing elements identified by their properties or location in the architecture. This focused domain allows FPath to offer a very concise and readable syntax and ensures correctness properties by construction (e.g. any query terminates in a finite time). *FScript* is a DSL dedicated to the reconfiguration of Fractal component architectures. It enables reconfiguration scripts to modify a Fractal architecture. Like FPath, FScript guarantees several properties by construction, e.g. termination of scripts by excluding the possibility of infinite loops. Moreover the FScript interpreter supports a transactional model of reconfigurations and the preservation of the ACID properties.

An adaptation of FPath/FScript to FraSCAti, a component framework providing runtime support for the Service Component Architecture (SCA), has been developed by the Inria Adam project-team. In that way, software architects are able to navigate using FPath notation through FraSCAti architectures and to reconfigure them with FScript. We have used this adaptation in our recent work [11][31] for reconfiguring cloud applications in order to reduce the energy footprint in cloud infrastructures.

FScript and its extensions are available under the LGPL license at http://fractal.ow2.org/fscript.

5.5. WildCAT

Participants: Thomas Ledoux [correspondent], Frederico Alvares.

monitoring, context-aware applications, complex event processing

WildCAT is a generic Java framework for context-aware applications. It permits the monitoring of large-scale applications by allowing developers to easily organize and access resources through a hierarchical organization backed with a powerful SQL-like language to inspect sensors data and to trigger actions upon particular conditions. WildCAT proposes two modes to inspect the resources: a pull mode relies on synchronous communication and a push one relies on asynchronous communication. In the pull mode, developers programmatically get and set attributes. In the push mode, developers register listeners on queries expressed over the events generated by the backend.

WildCAT has been developed by the team in the last years. We have used WildCAT in our recent work [11] for allowing cloud applications to listen events notification fired by the cloud infrastructure (e.g. whenever the pricing policy of cloud resources changes) or to detect changes on the application activity (e.g. to detect whenever the number of requests sharply increases/decreases) in order to launch the reconfiguration of cloud applications.

WildCAT is available under GPL v2 at http://wildcat.ow2.org.

6. New Results

6.1. Software composition

Participants: Akram Ajouli, Diana Allam, Omar Chebaro, Rémi Douence, Hervé Grall, Jean-Claude Royer, Mario Südholt.

We have produced results on service-oriented computing, language support for software composition, program transformation for composition, as well as the analysis of C programs.

6.1.1. Program transformation and formal properties

We have proposed an extension of the type theory underlying the Coq theorem prover and studied invertible transformations as a means to decompose object-oriented properties.

6.1.1.1. Forcing in the Calculus of Constructions and Coq

We have developed an intuitionistic forcing translation for the Calculus of Constructions (CoC), a translation that corresponds to an internalization of the presheaf construction in CoC [22]. Depending on the chosen set of forcing conditions, the resulting type theory can be extended with extra logical principles. The translation is proven correct—in the sense that it preserves type checking—and has been implemented in Coq. As a case study, we have shown how the forcing translation on integers (which corresponds to the internalization of the topos of trees) allows us to define general inductive types in Coq, without the strict positivity condition.

6.1.1.2. Invertible transformations for program decompositions

When one chooses a main axis of structural decomposition for a software, such as function- or data-oriented decompositions, the other axes become secondary, which can be harmful when one of these secondary axes becomes of main importance. In the context of modular maintenance, we have tackled this problem using invertible program transformations [19]. We have experimented our approach for Java [29] and Haskell programs.

In [29] we have presented such a transformation for Java. Precisely, we build a reversible transformation between Composite and Visitor design patterns in Java programs, based on chains of basic refactoring operations. Such transformations represent an automatic reversible switching between different program architectures with a guarantee of semantic preservation. The transformation is automated with the refactoring tool of a popular IDE: JetBrains Intellij Idea.

As seen in that paper, basic refactoring operations can be combined to perform complex program transformations. But the resulting composed operations are rarely reused, even partially, because popular tools have few support for composition. In [45] we have formalized the composition of refactoring operations of our Composite/Visitor transformation by the means of a static type system. That type system is based on two previous calculi for composition of refactoring, which we recast in one single calculus. The type system is used to prove non-failure and correctness of transformations. This kind of formalization yields a validation of transformations and, if integrated in existing IDEs, should help to reuse existing transformations.

6.1.2. Service-oriented computing

In the field of service-oriented computing, we have developed three contributions: a model for web services that enables WS*/SOAP-based heavyweight services and RESTful lightweight services to be handled uniformly, a type system that is safe in the presence of malicious agents and insecure communication channels, as well as a pivot language that provides a common abstraction for very different web query languages.

6.1.2.1. Uniform service model

Service-oriented applications are frequently used in highly dynamic contexts: service compositions may change dynamically, in particular, because new services are discovered at runtime. Moreover, subtyping has recently been identified as a strong requirement for service discovery. Correctness guarantees over service compositions, provided in particular by type systems, are highly desirable in this context. However, while service oriented applications can be built using various technologies and protocols, none of them provides decent support ensuring that well-typed services cannot go wrong. Currently, Service-Oriented Architecture applications are typically built using either the SOAP/WS or REST service models. Although there is a clear need for a model integrating both in multiple real-world contexts, no integrated model does (yet) exist. Therefore, in [15] we have introduced a model as a foundation for heterogeneous services, therefore unifying the SOAP/WSand RESTmodels.

6.1.2.2. A type system for services

We have presented a formal model in [14] for service compositions and defined a type system [33] with subtyping that ensures type soundness by combining static and dynamic checks. Our model allows channel mobility and inference of the type of discovered channels. This type system is based on the notion of semantic typing and proved to be sound. We have also demonstrated how to get type soundness in presence of malicious agents and insecure communication channels.

6.1.2.3. Criojo: a pivot language for services

Interoperability remains a significant challenge in service-oriented computing. After proposing a pivot architecture to solve three interoperability problems, namely adaptation, integration and coordination problems between clients and servers, we explore the theoretical foundations for this architecture. A pivot architecture requires a universal language for orchestrating services and a universal language for interfacing resources. Since there is no evidence today that Web Services technologies can provide this basis, we have proposed a new language called Criojo and shown that it can be considered as a pivot language. We have formalized the language Criojo and its operational semantics by resorting to a chemical abstract machine, and given an account of formal translations into Criojo: in a distributed context, we have dealt with idiomatic languages for four major programming paradigms: imperative programming, logic programming, functional programming and concurrent programming.

6.1.3. Languages and composition models

We have contributed new results in the domains of software product lines, model-based composition and language support for numerical constraint-based programming.

6.1.3.1. Software product lines and model composition

Many approaches to creating Software Product Lines have emerged that are based on Model-Driven Engineering. Our book [32] introduces both Software Product Lines and Model-Driven Engineering, which have separate success stories in industry, and focuses on the practical combination of them. It describes the challenges and benefits of merging these two software development trends and provides the reader with a novel approach and practical mechanisms to improve variability. Advanced concepts like fine-grained variability and decision models based on aspect-oriented programming techniques are illustrated. The concepts and methods are detailed with two product line examples: the classic smart-home systems and a collection manager information system.

6.1.3.2. Expressive language support for numerical constraint based programming

A combinatorial search can either be performed by using an implicit or an explicit search tree. We have proposed a functional DSL [35] for explicit search trees in the field of numerical constraints. The first advantage of our approach is expressiveness: we can write new algorithms or reformulate existing ones in a simple and unified way. The second advantage is efficiency, since an implicit search may also lead to a blowup of redundant computations. We illustrate this through various examples.

6.1.4. Analysis and test of C programs

Ascola members have participated, in cooperation with researchers from CEA List institute, in the development of analyses and corresponding tool support for C programs.

We have studied combinations of static and dynamic analysis techniques that enable the detection of out-of-bounds memory accesses in C programs and generate corresponding concrete test data [17]. This is particularly problematic for input arrays and pointers in C functions. We have presented a specific technique allowing the interpretation and execution of assertions involving the size of an input array (pointer) of a C function. We have successfully applied this technique in the Sante tool from the CEA where it allows potential out-of-bounds access errors to be detected and classified in several real-life programs.

PathCrawler is a test generation tool developed at CEA LIST for structural testing of C programs. The new version of PathCrawler [18] we have contributed to is developed in an entirely novel form: that of a test-case generation web service which is freely accessible at PathCrawler-online.com. This service allows many test-case generation sessions to be run in parallel in a completely robust and secure way. We have presented PathCrawler-online.com in the form of a lesson on structural software testing, showing its benefits, limitations and illustrating the usage of the tool on a series of examples.

6.2. Aspect-Oriented Programming

Participants: Rémi Douence, Guilhem Jaber, Ismael Mejía, Jacques Noyé, Mario Südholt, Nicolas Tabareau.

We have contributed to the foundations of aspect-oriented programming and presented new programming languages for aspects and related paradigms.

6.2.1. Formal models for AOP

We have presented two calculi contributing to the foundations of AOP: the A Calculus, a parameterized calculus encompassing AspectJ-like and history based aspect languages, and a category-theoretic definition of AOP in terms of 2-categories.

6.2.1.1. The A Calculus

With partners from Vrije Universiteit Brussel and Aarhus University, we have extended the foundational calculus for AOP (introduced in 2010) that supports the most general aspect model to-date compared to existing calculi and the deepest integration with plain OO concepts [12]. Integration with OOP is achieved essentially by modeling proceed using first-class closures. Two well-known pointcut categories, call and execution that are commonly considered similar are shown to be significantly different; our calculus enables the resolution of the associated soundness problems. The A-calculus also includes type ranges, an intuitive and concise alternative to explicit type variables that allows advices to be polymorphic over intercepted methods. Finally, our calculus is the first aspect calculus to use calculus parameters to cover type safety for a wide design space of other features. The soundness of the resulting type system has been verified in Coq.

In 2012, we have covered a range of choices with respect to evaluation order and non-determinism. We have studied one version that enforces a deterministic call-by-value semantics, and another one that omits restrictions on evaluation order and allows many kinds of non-determinism. Furthermore, we have provided a mechanized complete type soundness proof using the theorem prover Coq.

6.2.1.2. A category-theoretic foundation of aspects

Aspect-Oriented Programming (AOP) started fifteen years ago with the remark that modularization of so-called crosscutting functionalities is a fundamental problem for the engineering of large-scale applications. However, theoretical foundations of AOP have been much less studied than its applicability.

We have proposed [26] to put a bridge between AOP and the notion of 2-category to enhance the conceptual understanding of AOP. Starting from the connection between the λ-calculus and the theory of categories, we have defined an internal language for 2-categories and shown how it can be used to define the first categorical semantics for a realistic functional AOP language. We have then used this categorical framework to introduce the notion of computational 2-monads for AOP. We have illustrated their conceptual power by defining a 2-monad for Éric Tanter's execution levels—which constitutes the first algebraic semantics for execution levels—and then introducing the first exception monad transformer specific to AOP that gives rise to a non-flat semantics for exceptions by taking levels into account.

6.2.2. Programming languages for aspects and related paradigms

We have introduced three results related to aspect-based programming languages: an extension of EScala for multi-paradigm programming; Monascheme, a language for modular prototyping of aspect-based languages and language support for membranes, an aspect-based means for structuring computations.

6.2.2.1. Concurrent multi-paradigm programming with EScala

EScala integrates, around the notion of *declarative events*, object-oriented, aspect-oriented and event-based programming [30]. However, in spite of the fact that events naturally evoke some form of concurrency, there is no specific support for concurrency in EScala. It is up to the programmer to understand how to combine declarative events and Scala's support for concurrent programming. We have started working on injecting concurrency at the heart of declarative events so that events can indeed be naturally concurrent [28].

6.2.2.2. Monascheme: modular prototyping of aspect languages

We have then developed Monascheme [21], an extensible aspect-oriented programming language based on monadic aspect weaving. Extensions to the aspect language are defined as monads, enabling easy, simple and modular prototyping. The language is implemented as an embedded language in Racket. We illustrate the approach with an execution level monad and a level-aware exception transformer. Semantic variations can be obtained through monad combinations.

6.2.2.3. Structuring computations with aspect-based membranes

In most aspect-oriented languages, aspects have an unrestricted global view of computation. Several approaches for aspect scoping and more strongly encapsulated modules have been formulated to restrict the power of aspects. Our approach [27] leverages the concept of programmable membranes of Boudol, Schmitt and Stefani, as a means to tame aspects by customizing the semantics of aspect weaving locally. Membranes have the potential to subsume previous proposals in a uniform framework. Because membranes give structure to computation, they enable flexible scoping of aspects; because they are programmable, they enable visibility and safety constraints, both for the advised program and for the aspects. The power and simplicity of membranes open interesting perspectives to unify multiple approaches that tackle the unrestricted power of aspects.

6.3. Cloud applications and infrastructures

Participants: Frederico Alvares, Gustavo Bervian Brand, Yousri Kouki, Adrien Lèbre, Thomas Ledoux, Guillaume Le Louët, Jean-Marc Menaud, Jonathan Pastor, Rémy Pottier, Flavien Quesnel, Mario Südholt.

We have contributed on SLA management for Cloud elasticity, fully distributed and autonomous virtual machine scheduling, and energy-efficient Cloud infrastructures.

6.3.1. SLA Management for Cloud elasticity

In [23], we have introduced *CSLA*, the Cloud Service Level Agreement language. The CSLA language has been influenced by related work, in particular WSLA and SLA@SOI. It allows to describe the SLA between a cloud service provider and a cloud customer. One of the novelties of CSLA is that it integrates features dealing with QoS uncertainty and cloud fluctuations, such as *confidence*, *penalty* and *fuzziness*.

Cloud computing is a model for enabling on-demand access to a shared pool of configurable resources as services. However, the management of such elastic resources is a complex issue. In [24], we have proposed a SLA-driven approach for optimizing the resources capacity planning for Cloud applications. We have modeled Cloud services using a closed queuing network model and proposed an extension of a Mean Value Analysis (MVA) algorithm to take into account the concept of SLA. Then, based on capacity planning method, our solution calculates the optimal configuration of a Cloud application.

6.3.2. Fully distributed and autonomous virtualized environments

Extending previous preliminary results of the DVMS prototype, we have consolidated this system to obtain a fully distributed virtual machine scheduler [13]. This system makes it possible to schedule VMs cooperatively and dynamically in large scale distributed systems. Simulations (up to 64K VMs) and real experiments both conducted on the Grid'5000 large-scale distributed system [34] showed that DVMS is scalable. This building block is a first element of a more complete cloud OS, entitled DISCOVERY (DIStributed and COoperative mechanisms to manage Virtual EnviRonments autonomicallY) [66]. The ultimate goal of this system is to overcome the main limitations of the traditional server-centric solutions. The system, currently under investigation in the context of the Jonathan Pastor's PhD, relies on a peer-to-peer model where each agent can efficiently deploy, dynamically schedule and periodically checkpoint the virtual environments it manages.

6.3.3. Energy-efficient Cloud applications and infrastructures

As a direct consequence of the increasing popularity of Cloud Computing solutions, data centers are amazingly growing and hence have to urgently face with the energy consumption issue. Available solutions rely on Cloud Computing models and virtualization techniques to scale up/down application based on their performance metrics. Although those proposals can reduce the energy footprint of applications and by transitivity of cloud infrastructures, they do not consider the internal characteristics of applications to finely define a trade-off between applications Quality of Service and energy footprint. We have proposed a self-adaptation approach that considers both application internals and system to reduce the energy footprint in cloud infrastructure [31], [11]. Each application and the infrastructure are equipped with their own control loop, which allows them to autonomously optimize their executions. In addition, these autonomic loops are coordinated to avoid inconsistent states. This coordination improves the synergy between applications and infrastructure in order to optimize the infrastructure energy consumption [16].

We have extended our previous work on Entropy, a virtual machine placement manager, by the development of btrScript, a safe autonomic system for virtual machine management that includes actions and placement rules. Actions are imperative operations to reconfigure the data center and declarative rules specify the virtual machine placement. Administrators schedule both actions and rules, to manage their data center(s). They can also interact with the btrScript system in order to monitor the data center and compute the correct virtual machine placement [25]. btrScript and Entropy have been packaged in a common software btrCloud.

7. Bilateral Contracts and Grants with Industry

7.1. Cooperation with SIGMA group

Participants: Thomas Ledoux, Simon Dupont.

In 2012, we have started a two-fold 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 amazingly growing and hence have to urgently face with the energy consumption issue. 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.

8. Partnerships and Cooperations

8.1. National Initiatives

8.1.1. ANR

8.1.1.1. CESSA: Compositional Evolution of Secure Services with Aspects (ANR/ARPEGE)

Participants: Mario Südholt [coordinator], Diana Allam, Rémi Douence, Hervé Grall, Jean-Claude Royer.

The project CESSA is an (industrial) ANR project running for 3 years months, with funding amounting to 290 KEUR for ASCOLA from Jan. 10 on. Three other partners collaborate within the project that is coordinated by ASCOLA: a security research team from Eurecom, Sophia-Antipolis, the Security and Trust team from SAP Labs, also located at Sophia-Antipolis, and IS2T, an innovative start-up company developing middleware technologies located at Nantes. The project deals with security in service-oriented architectures.

This year our group has contributed several scientific publications as part of the project. All partners have been involved in the publication of a unifying model for WD*/SOAP-based and RESTful web services. Furthermore, we have formally defined a type system that is safe in the presence of malicious attackers and insecure communication channels.

All information is available from the CESSA web site: http://cessa.gforge.inria.fr.

8.1.1.2. Entropy (ANR/Emergence)

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

The Entropy project is an (industrial) ANR/Emergence project running for 18 months. It was accepted in December 2010 for funding amounting to 242 KEUR (ASCOLA only).

The objective of this project is to conduct studies on economic feasibility (market, status, intellectual property, website) for creating a industrial business on the Entropy software.

Some task must complete the Entropy core solution with a graphical unit interface to produce convincing demonstrators and consolidate our actual and future results. At the end of the project, the goal is to create a company whose objective is to sell the service, support and software building blocks developed by this ANR Emergence project.

8.1.1.3. MyCloud (ANR/ARPEGE)

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

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.

This year, the ASCOLA project-team has proposed (i) CSLA, a novel language to describe QoS-oriented SLA associated with cloud services [23]; (ii) a SLA-driven capacity planning for cloud applications [24].

8.1.1.4. SONGS (ANR/INFRA)

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

The SONGS project (Simulation of Next Generation Systems) is an ANR/INFRA project running for 48 months (starting from 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. Each type of large-scale computing system will be addressed through a set of use cases and lead by researchers recognized as experts in this area. The ASCOLA involvement will start in 2013 with the arrival of Takahiro Hirofuchi from the AIST institute in Japan.

8.1.2. FUI

8.1.2.1. Cool-IT (FUI)

Participant: Jean-Marc Menaud [coordinator].

The Cool-IT project is an (industrial) FUI project running for 24 months. It was accepted in September 2010 for funding amounting to 130 KEUR (ASCOLA only).

The objective of this project is to design systems adapted to new standards of "Green IT" to reduce the data centers electrical consumption.

To this end, the COOL IT project will develop processes for cooling computer servers, optimize the server power chain supply, implement tools and methods of collecting energy data in real time, and specify methods for controlling the data centers consumption as a tradeoff between the computational power needed, the availability, and the energy consumption.

8.1.3. FSN

8.1.3.1. 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 accross Cloud layers) and "Run" (virtual machine manager for datacenters and placement constraints).

8.2. European Initiatives

8.2.1. FP7 Projects

8.2.1.1. A4Cloud: Accountability for the Cloud (Integrated Project)

Participants: Mario Südholt [coordinator], Omar Chebaro, Ronan-Alexandre Cherrueau, Rémi Douence, Hervé Grall, Jean-Claude Royer.

The A4Cloud project is an integrated EU project, coordinated by HP, UK, on the topic of accountability, that is, the responsible stewardship of private data, in the Cloud. This 42-months project started in Oct. 2012 and Ascola's funding amounts to 600 KEuro.

The project involves 13 partners: in addition to HP, two enterprises (SAP AG, Germany; ATC, Greece), a non-governmental organisation (the Cloud Security Alliance, CSA) and 9 universities and research organisations (EMNantes and Eurecom, France; HFU. Furtwangen, Germany; Karlstadt U., Sweden; U. Malaga, Spain; Queen Mary U., U.K.; U. Stavanger and Sintef, Norway; Tilburg U., The Netherlands).

A4Cloud will create solutions to support users in deciding and tracking how their data is used by cloud service providers. By combining methods of risk analysis, policy enforcement, monitoring and compliance auditing with tailored IT mechanisms for security, assurance and redress, A4Cloud aims to extend accountability across entire cloud service value chains, covering personal and business sensitive information in the cloud.

8.2.2. Collaborations in European Programs, except FP7

8.2.2.1. SCALUS: SCALing by means of Ubiquitous Storage (MC ITN)

Participants: Adrien Lèbre [coordinator], Mario Südholt, Gustavo Bervian Brand.

The vision of the Scalus Marie Curie international training network (MC ITN) is to deliver the foundation for ubiquitous storage systems, which can be scaled with respect to multiple characteristics (capacity, performance, distance, security, ...).

Providing ubiquitous storage will become a major demand for future IT systems and leadership in this area can have significant impact on European competitiveness in IT technology. To get this leadership, it is necessary to invest into storage education and research and to bridge the current gap between local storage, cluster storage, grid storage, and cloud storage. The consortium will proceed into this direction by building the first interdisciplinary teaching and research network on storage issues. It consists of top European institutes and companies in storage and cluster technology, building a demanding but rewarding interdisciplinary environment for young researchers.

The network involves the following partners: University of Paderborn (Germany, coordinator), Barcelona Super Computing (Spain), University of Durham (England), University of Frankfurt (Germany), ICS-FORTH (Greece), Universidad Polytecnica de Madrid (Spain), EMN/ARMINES (France), Inria Rennes Bretagne Atlantique (France), XLAB (Slovenia), University of Hamburg (Germany), Fujistu Technology Systems (Germany).

The overall funding of the project by the European Union is closed to 3,3 MEUR. ASCOLA's share amounts to 200 KEUR.

8.2.2.2. COST IC0804

Program: Energy efficiency in large sclae distributed systems

Project acronym: COST IC0804

Project title: Energy efficiency in large scale distributed systems

Duration: Jan. 2009 - May 2013

Coordinator: Jean-Marc Pierson (IRIT, France)

Participating countries: AT, BE, CH, CY, DE, DK, EE, FI, FR, GR, HU, IE, IL, IT, LU, PL, PT, RO, SE, SP, TR, UK,

Abstract: The COST IC 0840 Action will propose realistic energy-efficient alternate solutions to share IT distributed resources. As large scale distributed systems gather and share more and more computing nodes and storage resources, their energy consumption is drastically increasing. While much effort is nowadays put into hardware specific solutions to lower energy consumptions, the need for a complementary approach is necessary at the distributed system level, i.e. middleware, networks and applications. The action will characterize the energy consumption and energy efficiencies of distributed applications. http://www.cost804.org/

8.3. International Initiatives

8.3.1. Inria Associate Teams

8.3.1.1. RAPIDS

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

Inria principal investigator: Jacques Noyé

International Partner (Institution - Laboratory - Researcher):

University of Chile (Chile) - PLEIAD - Éric Tanter

Duration: 2010 - 2012

See also: http://rapids.gforge.inria.fr/doku.php

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. International Research Visitors

8.4.1. Internships

Rahma CHAABOUNI (from April 2012 until June 2012)

Subject: Flexible evolution of service-oriented systems

Institution: ENIS school, Sfax, Tunisie

Ismael FIGUEROA (from May 2012 until Jul 2012)

Subject: Exploring membranes for aspect oriented programming

Institution: University of Chile (Chile)

9. Dissemination

9.1. Scientific Animation

9.1.1. Event organization, animation of scientific community

9.1.1.1. International

- **Aspect-oriented Software Association (AOSA):** Mario Südholt has been the chair of this international organization (headquarted in California) since 2011
- **COST Action:** Jean-Marc Menaud is a management committee member of the European COST Action IC0804: Energy efficiency in large scale distributed systems from 2009 to 2013.
- **Grid'5000 Winter School.** A. Lèbre has been the general chair of the 5th edition of the Grid'5000 winter school organized by EMN, the LINA laboratory and Inria in Nantes, December 3-6 2012 (70 attendees). https://www.grid5000.fr/mediawiki/index.php/Grid5000:School2012.
- **OW2:** Jean-Marc Menaud has been a member of the OW2 board committee since 2010.

9.1.1.2. National

- **COMIN Labs laboratory of Excellence:** Jean-Marc Menaud has been the co-coordinator of the focus on Energy and resource efficiency in ICT since Jun. 2011. Mario Südholt has been co-coordinating the Security focus of COMIN Labs.
- **Enterprise Information Systems in the Cloud.** M. Südholt has been part of the organization committee of this event organized by the Images&Networking competitivity cluster in Sep, 2012. He has also given an invited lecture there on the foundations of the Cloud.
- **Eco-conception of software.** J.-M. Menaud co-organized the first event on *Eco-conception des logiciels* in Nantes, and an event on *Efficacité énergétique : comment monitorer et quels sont les impacts sur les applications ?* in Lyons.
- **Software Product Line series of events.** J.-C. Royer started a steering committee for the *Journée lignes de produits logiciels*, which took place in Lille, November 6, http://www.jldp.org/2012

9.1.2. Committee participations, reviewing activities, collective duties, etc.

- **P. Cointe:** He is the head of the LINA computer science laboratory (UMR 6241)http://www.lina.univ-nantes.fr/. As such he was involved in the CominLabs (labex) proposal.

  He is a member of the board of the Doctoral School MATISSE in Rennes as well as the selection and evaluation committee and the board of the cluster Images & Réseaux. He is the chair of the STIC-MATHS committee working for the Pays de la Loire Council (CCRRDT) and the PRES L'UNAM.

- **A. Lèbre** has been the program chair of the 6th international workshop on "Virtualization Technologies in Distributed Computing" (VTDC 2012) co-located with HPDC 2012 in Delft, the Netherlands. He also was a member of the program committee of the IEEE International Conference on Cloud and Green Computing (CGC 2012). He also took part to several reviewing processes for the IEEE Transactions on Parallel and Distributed Systems Journal, the Concurrency and Computation: Practice and Experience Journal, the French journal *Technique des Sciences Informatiques* and has been involved as external reviewers for the ISPA, the ICPADS and the Europar conferences.
- **T. Ledoux** was a member of the program committees of the following conferences: 3rd International Workshop on Green and Cloud Management (GCM'12), 11th International Workshop on Adaptive and Reflective Middleware (ARM'12), Second International Conference on ICT as Key Technology against Global Warming (ICT- GLOW'12), 28th Symposium On Applied Computing (SAC'13) - track Software Engineering Aspects of Green Computing.

  He is a member of the board of the Regional Doctoral School STIM. He is a member of the board of the Follow-up committee of the PhD theses at LINA.
- **J.-M. Menaud** has served on the program committee of the 7th Workshop on Virtualization in High-Performance Cloud Computing (VHPC'12), the first International Workshop on Cloud and Grid Interoperability (Cloud&Grid 2012), the IEEE/ACM Inter. Conference on Green Computing and Communications (GreenCom2012), the Eighth Advanced International Conference on Telecommunications (AICT 2012), the Second International Conference on Smart Grids, Green Communications and IT Energy-aware Technologies (ENERGY 2012), the first International Conference on Smart Grids and Green IT Systems (SMARTGREENS 2012), the second International Conference on ICT as Key Technology against Global Warming (ICT-GLoW 2012), the first Workshop on Collaborative and Autonomic Green computing (CAGing 2012) and the French Conference *NOuvelles Technologies de la REpartition* (NOTERE'12). He was a reviewer for the International Journal of Computer and Telecommunications Networking (COMNET), the International Journal of the Electronics and Telecommunications Research Institute, and the International Journal on Transactions on Parallel and Distributed Systems.

  J.-M. Menaud is a management committee member of the european COST Action IC0804 : Energy efficiency in large scale distributed systems since 2009, the co-animator of the focus Energy and resource efficiency in ICT, in the Labex COMIN Labs since June 2011, the animator of the CNRS/GDR ASR System group since June 2009, member of the OW2 board committee from 2010 and a member of the (RenPar/CFSE/Sympa) steering committee since 2008.
- **J. Noyé** co-organized the 7th international workshop on "Domain-Specific Aspect Languages" (DSAL 2012) co-located with AOSD 2012 in Potsdam, Germany.
- **J-C. Royer:** He is a member of the editorial board of the journal TSI. He was a member of the program committee of CAL 2012 and CIEL 2012 and did reviews for the SPLC conference, and the RNTI and SoSym journals.
- **M. Südholt** is the chair of the international associtation of AOSD. He is a member of the steering committees of the international conferences AOSD and Software composition. He is also a member of the editorial board of the international journal "Transactions of AOSD", which is published by Springer Verlag.

  He was a member of the program committes of the international conferences GPCE'12 and AOSD'12 as well as of 2 international and 2 national workshops.

  He is a member of the council of the *Laboratoire Informatique de Nantes Atlantique* (LINA, UMR 6241). He also serves on the selection and evaluation committee of the competitiveness cluster Images & Réseaux. Finally, he is a member of the governing board of AOSD-Europe, the European Network of Excellence in AOSD.

9.2. Teaching - Supervision - Juries

9.2.1. Teaching

The team is involved in the following undergraduate and graduate-level programs at EMN (the institution 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 final-year **graduate-level informatics specialization**.
- Since 2009 our team has defined and set up a new three-year **engineering program on software engineering**

The team has also been involved involved in the following three 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.
- 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.

m members have taught each for about 190 hours on average in 2012 (hours of presence in front of students). Hereby, we have taken into account that researchers and some professors have not taught at times and that part of the program is taught by temporary staff.

9.2.2. Supervision

The team has been supervising 16 PhD thesis in 2012, of which four have been co-supervised with external partners (three with foreign partners from U. Chile; TU Darmstadt, Germany; Lancaster U., U.K.) and one with the French TASC team from EMNantes.

One PhD has been defended in Sep. 2012, Remy Pottiers thesis on "Approche langage pour l'administration d'infrastructures virtualisées".

Two members of the team have started preparing an HDR in 2012 for defense in 2013.

9.2.3. Juries

- **A. Lèbre** was member of the PhD committee of Sheheryar Malik (University of Nice).
- **J.-M. Menaud** was a reviewer of the PhD of Sheheryar Malik, Nice University, and member of the PhD committees of Xavier Etchevers, Grenoble University, and Eugen Feller, Rennes University.
- **J-C. Royer** was a reviewer of the PhD of Alfred Sanogo, Université Paris Nord, and a member of the PhD committees of Rémy Pottier, EMN, and Takoua Ben-Rouhma, Université Paris Sud.
- **M. Südholt** was a reviewer of the PhD of Arjan De Roo, university of Twente, the Netherlands.

10. Bibliography

Major publications by the team in recent years

[1] F. ALVARES DE OLIVEIRA JR., R. SHARROCK, T. LEDOUX. *Synchronization of Multiple Autonomic Control Loops: Application to Cloud Computing*, in "COORDINATION - International Conference on Coordination Models and Languages - 2012", Stockholm, Sweden, March 2012, http://hal.inria.fr/hal-00682914.

[2] 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.

[3] F. HERMENIER, X. LORCA, J.-M. MENAUD, G. MULLER, J. LAWALL. *Entropy: a Consolidation Manager for Clusters*, in "The ACM SIGPLAN/SIGOPS Int. Conference on Virtual Execution Environments (VEE'09)", March 2009.

[4] G. JABER, N. TABAREAU, M. SOZEAU. *Extending Type Theory with Forcing*, in "LICS 2012 : Logic In Computer Science", Dubrovnik, Croatia, June 2012, http://hal.inria.fr/hal-00685150.

[5] M. LÉGER, T. LEDOUX, T. COUPAYE. *Reliable Dynamic Reconfiguration in a Reflective Component Model*, in "Proc. of the 13th Int. Symposium on Component Based Software Engineering (CBSE'10)", Tchèque, République, Lecture Notes in Computer Science, Springer-Verlag, June 2010, p. 74-92.

[6] F. QUESNEL, A. LÈBRE, 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] P. RITEAU, A. LÈBRE, C. MORIN. *Handling Persistent States in Process Checkpoint/Restart Mechanisms for HPC Systems*, in "Proceedings of the 9th IEEE International Symposium on Cluster Computing and Grid (CCGRID 2009)", Shangai, China, IEEE Computer Society Press, 2009.

[8] N. TABAREAU. *A theory of distributed aspects*, in "9th International Conference on Aspect-Oriented Software Development (AOSD '10)", France Rennes, Saint-Malo, ACM, 2010, p. 133–144, http://dx.doi.org/10.1145/1739230.1739246.

[9] É. TANTER, J. FABRY, R. DOUENCE, J. NOYÉ, M. SÜDHOLT. *Scoping strategies for distributed aspects*, in "Science of Computer Programming", July 2010, http://dx.doi.org/10.1016/j.scico.2010.06.011.

[10] 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.

Publications of the year

Articles in International Peer-Reviewed Journals

[11] F. ALVARES DE OLIVEIRA JR., T. LEDOUX. *Self-management of cloud applications and infrastructure for energy optimization*, in "SIGOPS Operating Systems Review", 2012, vol. 46, n^{o} 2, http://hal.inria.fr/hal-00710695.

[12] 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.

[13] F. QUESNEL, A. LÈBRE, 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.

International Conferences with Proceedings

[14] D. ALLAM. *A unified formal model for service oriented architecture to enforce security contracts*, in "Proceedings of the 11th annual international conference on Aspect-oriented Software Development Companion", New York, NY, USA, AOSD Companion '12, ACM, 2012, p. 9–10, http://doi.acm.org/10.1145/2162110.2162120.

[15] D. ALLAM, R. DOUENCE, H. GRALL, J.-C. ROYER, M. SÜDHOLT. *A Message-Passing Model for Service Oriented Computing*, in "WEBIST, 8th International Conference on Web Information Systems and Technologies", Porto, Portugal, K.-H. KREMPELS, J. CORDEIRO (editors), SciTePress Digital Library, April 2012, http://hal.inria.fr/hal-00668975.

[16] F. ALVARES DE OLIVEIRA JR., R. SHARROCK, T. LEDOUX. *Synchronization of Multiple Autonomic Control Loops: Application to Cloud Computing*, in "COORDINATION - International Conference on Coordination Models and Languages - 2012", Stockholm, Sweden, March 2012, http://hal.inria.fr/hal-00682914.

[17] O. CHEBARO, M. DELAHAYE, N. KOSMATOV. *Testing Inexecutable Conditions on Input Pointers in C Programs with SANTE*, in "24th International Conference on Software & Systems Engineering and their Applications (ICSSEA 2012)", Paris, France, October 2012, 7, http://icssea.enst.fr/icssea12/, http://hal.inria.fr/hal-00724508.

[18] O. CHEBARO, N. KOSMATOV, N. WILLIAMS, B. BOTELLA, M. ROGER. *A lesson on structural testing with PathCrawler-online.com*, in "6th International Conference on Tests & Proofs", Prague, Czech Republic, June 2012, 6 pages, http://hal.inria.fr/hal-00685504.

[19] J. COHEN, R. DOUENCE, A. AJOULI. *Invertible Program Restructurings for Continuing Modular Maintenance*, in "16th European Conference on Software Maintenance and Reengineering (CSMR 2012)", Szeged, Hungary, R. F. TOM MENS (editor), IEEE, March 2012, p. 347–352, 6 pages, Early Research Achievements Track [*DOI :* 10.1109/CSMR.2012.42], http://hal.inria.fr/hal-00662777.

[20] T. DINKELAKER, J. FABRY, J. NOYÉ. *Proceedings of the seventh workshop on Domain-Specific Aspect Languages (DSAL 2012)*, in "AOSD - Aspect-Oriented Software Development - 2012", Potsdam, Germany, A. PRESS (editor), March 2012, http://hal.inria.fr/hal-00726770.

[21] I. FIGUEROA, É. TANTER, N. TABAREAU. *A Practical Monadic Aspect Weaver*, in "Foundations of Aspect-Oriented Languages", Potsdam, Germany, March 2012, http://hal.inria.fr/hal-00690717.

[22] G. JABER, N. TABAREAU, M. SOZEAU. *Extending Type Theory with Forcing*, in "LICS 2012 : Logic In Computer Science", Dubrovnik, Croatia, June 2012, http://hal.inria.fr/hal-00685150.

[23] Y. KOUKI, T. LEDOUX. *CSLA : a Language for Improving Cloud SLA Management*, in "International Conference on Cloud Computing and Services Science, CLOSER 2012.", Porto, Portugal, April 2012, http://hal.inria.fr/hal-00675077.

[24] Y. KOUKI, T. LEDOUX. *SLA-driven Capacity Planning for Cloud applications*, in "IEEE International Conference on Cloud Computing Technology and Science (CloudCom 2012)", Taipei, Taiwan, Province Of China, December 2012, http://hal.inria.fr/hal-00734417.

[25] J.-M. MENAUD, R. POTTIER. *btrScript : a safe management system for virtualized data center*, in "The 8th International Conference on Autonomic and Autonomous Systems (ICAS 2012)", XPS, March 2012, http://hal.inria.fr/hal-00656091.

[26] N. TABAREAU. *A Monadic Interpretation of Execution Levels and Exceptions for AOP*, in "Modularity: AOSD'12", Postdam, Germany, ACM Press, March 2012, http://hal.inria.fr/inria-00592132.

[27] É. TANTER, N. TABAREAU, R. DOUENCE. *Taming Aspects with Membranes*, in "Foundations of Aspect-Oriented Languages", Potsdam, Germany, March 2012, http://hal.inria.fr/hal-00690706.

[28] J. M. VAN HAM. *Adding high-level concurrency to EScala*, in "Proceedings of the 11th annual international conference on Aspect-oriented Software Development Companion", New York, NY, USA, AOSD Companion '12, ACM, 2012, p. 19–20, http://doi.acm.org/10.1145/2162110.2162125.

National Conferences with Proceeding

[29] A. AJOULI. *An Automatic Reversible Transformation from Composite to Visitor in Java*, in "CIEL 2012", P. Collet, P. Merle (eds.); Conférence en IngénieriE du Logiciel (CIEL), June 2012, http://hal.inria.fr/hal-00733182.

[30] J. NOYÉ. *E{Java, CaesarJ, Scala} : un exercice d'intégration de la programmation par objets, par aspects et par évènements*, in "Quatrièmes journées nationales du GDR GPL", Rennes, France, L. DUCHIEN, O. BARAIS (editors), June 2012, p. 85-86, http://hal.inria.fr/hal-00726618.

Scientific Books (or Scientific Book chapters)

[31] F. ALVARES DE OLIVEIRA JR., A. LÈBRE, T. LEDOUX, J.-M. MENAUD. *Self-management of applications and systems to optimize energy in data centers*, in "Achieving Federated and Self-Manageable Cloud Infrastructures: Theory and Practice", I. BRANDIC, M. VILLARI, F. TUSA (editors), IGI Global, February 2012, http://hal.inria.fr/hal-00670033.

[32] H. ARBOLEDA, J.-C. ROYER. *Model-Driven and Software Product Line Engineering*, ISTE LTd and John Wiley & Sons, Inc., August 2012, 288, http://hal.inria.fr/hal-00734143.

Research Reports

[33] D. ALLAM, R. DOUENCE, H. GRALL, J.-C. ROYER, M. SÜDHOLT. *Well-Typed Services Cannot Go Wrong*, Inria, May 2012, n^{o} RR-7899, http://hal.inria.fr/hal-00700570.

[34] D. BALOUEK, A. CARPEN AMARIE, G. CHARRIER, F. DESPREZ, E. JEANNOT, E. JEANVOINE, A. LÈBRE, D. MARGERY, N. NICLAUSSE, L. NUSSBAUM, O. RICHARD, C. PÉREZ, F. QUESNEL, C. ROHR, L. SARZYNIEC. *Adding Virtualization Capabilities to Grid'5000*, Inria, July 2012, n^{o} RR-8026, 18, http://hal.inria.fr/hal-00720910.

[35] G. CHABERT, R. DOUENCE. *From Implicit to Explicit Pavings*, Inria, July 2012, n^{o} RR-8028, http://hal.inria.fr/hal-00720739.

Other Publications

[36] H. GRALL, M. LACOUTURE. *Criojo: A Pivot Language for Service-Oriented Computing - The Introspective Chemical Abstract Machine*, 2012, Internal report, http://hal.inria.fr/hal-00676083.

References in notes

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

[38] 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.

[39] 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, p. 213–49.

[40] 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, p. 187–209.

[41] 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, p. 51-62.

[42] 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.

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

[44] E. M. CLARKE, O. GRUMBERG, D. A. PELED. *Model Checking*, The MIT Press, Cambridge, Massachusetts, 1999.

[45] J. COHEN, A. AJOULI. *Practical use of static composition of refactoring operations*, in "ACM Symposium On Applied Computing", Portugal, March 2013, 6, http://hal.archives-ouvertes.fr/hal-00751304.

[46] M. COLE. *Algorithmic Skeletons: Structured Management of Parallel Computation*, MIT Press, 1989.

[47] 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, p. 56–65.

[48] 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, p. 80-86.

[49] 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, p. 296–303.

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

[51] 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, p. 173–188, http://hal.inria.fr/inria-00072153.

[52] 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, p. 201-218.

[53] 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, p. 170–186.

[54] 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, p. 79-88.

[55] P. T. EUGSTER, P. A. FELBER, R. GUERRAOUI, A.-M. KERMARREC. *The many faces of publish/subscribe*, in "ACM Computing Surveys", June 2003, vol. 35, n^{o} 2, p. 114–131, http://doi.acm.org/10.1145/857076.857078.

[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, p. 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, p. 342–361.

[58] E. GAMMA, R. HELM, R. JOHNSON, J. VLISSIDES. *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison Wesley, Massachusetts, 1994.

[59] 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, p. 273–284, http://www.doc.ic.ac.uk/~yoshida/multiparty/multiparty.pdf, http://doi.acm.org/10.1145/1328438.1328472.

[60] 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, Berlin, June 2001, vol. 2072, p. 327–353, http://www.eclipse.org/aspectj/.

[61] 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.

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

[63] 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.

[64] 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, p. 197–214.

[65] X. LEROY. *Manifest types, modules, and separate compilation*, in "Manifest types, modules, and separate compilation", Portland, Oregon, USA, ACM Press, January 1994, p. 109-121.

[66] A. LÈBRE, P. ANEDDA, M. GAGGERO, F. QUESNEL. *DISCOVERY, Beyond the Clouds - DIStributed and COoperative framework to manage Virtual EnviRonments autonomicallY: a prospective study*, in "Virtualization for High Performance Cloud Computing workshop (colocated with EUROPAR 2011)", Bordeaux, France, August 2011, http://hal.inria.fr/hal-00645912/en.

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

[68] 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, p. 70-93.

[69] N. R. MEHTA, N. MEDVIDOVIC, S. PHADKE. *Towards a Taxonomy of Software Connectors*, in "Proceedings of ICSE", Limerick, Ireland, jun 2000, p. 178–187.

[70] 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, p. 316-344.

[71] 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, p. 355-382.

[72] R. T. MONROE, A. KOMPANEK, R. MELTON, D. GARLAN. *Architectural Styles, Design Patterns, and Objects*, in "IEEE Software", January 1997, vol. 14, n^{o} 1, p. 43-52.

[73] 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.

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

[75] 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.

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

[77] F. PLASIL, S. VISNOVSKY. *Behavior Protocols for Software Components*, in "Transactions on Software Engineering", January 2002, vol. 28, n^{o} 9.

[78] 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, p. 367–388.

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

[80] B. C. SMITH. *Reflection and Semantics in LISP*, Xerox Palto Alto Research Center, Palo Alto, 1984, n^{o} P84-00030.

[81] 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, p. 174–190.

[82] 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, p. 159 - 169.

[83] 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^{o} 5, p. 890–910.

[84] D. M. YELLIN, R. E. STROM. *Protocol specifications and component adaptors*, in "ACM Transactions of Programming Languages and Systems", March 1997, vol. 19, n^{o} 2, p. 292–333.

[85] A. VAN DEURSEN, P. KLINT, J. VISSER. *Domain-Specific Languages: An Annotated Bibliography*, in "ACM SIGPLAN Notices", June 2000, vol. 35, n^{o} 6, p. 26-36.