IN PARTNERSHIP WITH:
CNRS

Ecole des Mines de Nantes

Activity Report 2011

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

Project-Team ASCOLA

Keywords: Domain-Specific Languages, Energy Consumption, Formal Methods, Aspect Oriented Programming, Cloud Computing

1. Members

Research Scientists

Rémi Douence [Junior Researcher Inria on leave from EMN since Sep. 10]
Nicolas Tabareau [Junior Researcher Inria]

Faculty Members

Tony Bourdier [Associate Professor (temporary), since Oct. 10]
Pierre Cointe [Professor, EMN, HdR]
Hervé Grall [Associate Professor, EMN]
Fabien Hermenier [Associate Professor (temporary), until Sep. 10]
Adrien Lèbre [Associate Professor, EMN]
Thomas Ledoux [Associate Professor, EMN]
Christine Louberry [Associate Professor (temporary), until Sep. 10]
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), since Oct. 10]
Mario Südholt [Team Leader: Professor, EMN, HdR]

Technical Staff

Thomas Chavrier [Inria (ADT Vasp), until Aug. 11]
Clotilde Massot [Inria (ADT Vasp), since Sep. 11]
Thierry Bernard [EMN (ANR Emergence Entropy) grant, since Jun. 11]
Frédéric Dumont [EMN (ANR Emergence Entropy) grant, since Oct. 11]

PhD Students

Akram Ajouli [EMN grant, since Oct. 10]
Diana Allam [ANR Cessa grant, since Oct. 10]
Frederico Alvares [MESR grant, since Oct. 09]
Ali Assaf [MESR grant, until Oct. 11]
Gustavo Bervian Brand [EU MCITN Scalus grant, since Sep. 10]
Abdelhakim Hannousse [Regional council & EMN grant, until Nov. 11]
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]
Ismael Mejía [EMN grant, since Jan. 09]
Syed Asad Ali Navqi [EMN & Univ. of Lancaster grant, since Nov. 08]
Dong Ha Nguyen [MESR grant, until Oct. 11]
Hien Nguyen Van [Cifre Orange grant, until Feb. 11]
Angel Núñez [EMN & IST Ample & Ater polytech grant, until Jun. 11]
Rémy Pottier [ANR Selfxl grant, since Apr. 09]
Flavien Quesnel [EMN grant, since Oct. 09]
Jurgen Van Ham [Armines grant, since Sep. 10]

Administrative Assistants

Diana Gaudin [Part-time (33%)]

Cécile Derouet [Part-time (33%), since Oct. 11]
Hanane Maaroufi [Part-time (50%), until Aug. 11]

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

This year the ASCOLA team has produced several particularly notable results in the academic realm as well as concerning the transfer of academic results to industry.

We have contributed several first-rate results on new programming language features. We have shown, in particular, how to define the Java security model using aspects [21], how to integrate Object-Oriented programming, Event-Based Programming and Aspect-Oriented Programming [24], and have extended results on the preservation of formal properties in the presence of aspects [17] (see Sec. 6.1 for details).

Based on ASCOLA's results on virtual machine management (developed under the lead of Jean-Marc Menaud), the start-up EasyVirt was founded in July 2011. EasyVirt develops and distributes software solutions for the management and optimization of data centers in active cooperation with the ASCOLA team (see Sec. 7.1).

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 [78]. 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 [87], [60]. 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 [77].

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. Such 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 [82].

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* [90] of a software system describes the system as a composition of *components* and *connectors*, where the connectors capture the *interaction protocols* between the components [52]. It also describes the rationale behind such a given architecture, linking the properties required from the system to its implementation. *Architecture Descriptions Languages* (ADLs) are languages that support architecture-based development [79]. 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* [62] 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 [91], [74] 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 [73], [50] 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 [72], 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 [92]. 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 [59]. 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 [94]) 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* [63], [64], 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 [86], [54], [66], provide more concise formal semantics for aspects and enable analysis of their properties [53], [65], [63], [51]. 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 [85], component interactions [95], [88] and service orchestrations [61] 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 [57], [68].

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 [71], or context-free or turing-complete expressiveness [89], [56]. 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 [63] and non-regular ones [93], [84]. 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 [83], connectors between (complex) software artifacts [80], or (based on, possibly concretizations of, *design patterns* [70]) 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* [67] as well as the large set of so-called *skeletons* [58], 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 [81], [96].

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* [55], [76]. 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 [75] and the modularization of the distribution concern itself [92]. 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 [69] 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 [53], [66].

4. Application Domains

4.1. Enterprise Information Systems

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.

This year, we have shown, in particular, how to use invasive patterns to reconcile requirements for black-box and gray-box composition for the evolution of grid-based information systems, see Sec. 6.1 and 5.1.

4.2. Service-Oriented Architectures

Service-Oriented Computing (SOC) is an emerging paradigm used to program and integrate distributed applications, thus 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.

4.3. 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 consists 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).

This year, we have investigated in particular, how we can manage more and more virtual machines (VMs) in IaaS (Infrastucture as a Service) infrastructures, taking into account the needs of the applications, VM constraints and energy consideration, see Sec. 6.3.

We continued to deal with self-administration of large distributed architectures with the partners of the ANR SelfXL project, see Sec. 8.1. Furthermore, we finalized the MCITN Scalus European network in which we are involved. The project SCALUS (SCALing by means of Ubiquitous Storage) addresses the foundation for ubiquitous storage systems, which can be scaled with respect to multiple characteristics (capacity, performance, distance, security, ...), see Sec. 8.2. We are involved in a new ANR project, entitled SONGS (Simulation Of Next Generation Systems). This project aims at modeling different distributed systems to deliver an accurate framework for simulations. We are going to address all virtualization related concerns.

4.4. 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.

These kinds of systems provide good case studies for our languages combining advanced features inherited from Object-Oriented, Aspect-Oriented, and Event-based Programming (see Sec. 5.2).

5. Software

5.1. AWED

Participants: Mario Südholt [correspondent], Ismael Mejia.

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.

This year a gray-box model for distributed composition has been built and implemented based on the AWED model using the notion of invasive distributed patterns (see Sec. 6.1). Furthermore, the resulting model has been applied to the evolution of grid applications and OpenMRS, an open-source health information system. 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. Finally, the development of a new, more modular, implementation of the AWED system has started in 2011.

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

5.2. ECaesarJ, EJava and EScala

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

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/.

5.3. Entropy

Participants: Jean-Marc Menaud [correspondent], Fabien Hermenier, Adrien Lèbre, Hien Nguyen Van, Rémy Pottier, Thomas Chavrier, Guillaume Le Louët.

Entropy is a virtual machine (VM) manager for clusters. The current prototype acts as an infinite control loop, which performs a globally optimized placement according to cluster resource usage, scheduler objectives and administrative rules.

Relying on an encapsulation of jobs into VMs, Entropy enables the implementation of finer scheduling policies through cluster-wide context switches, i.e., permutations of VMs present in the cluster. It thus supports a more flexible use of cluster resources and frees end-users from the burden of dealing with time estimates.

The major advantage of the Entropy system concerns the cluster-wide context switch process itself. Entropy computes a new viable configuration and an optimized reconfiguration plan. This plan describes the sequences of transitions to perform (i.e. the run, migrate, suspend/resume, stop VM operations) in order to transit from the current situation to the new one. As the cost of each action and the dependencies between them is considered, Entropy reduces the duration of each cluster-wide context switch by performing a minimum number of actions in the most efficient way.

Around this solution, we developed VMScript, a domain-specific language for administration of virtualized grid infrastructures. This language relies on set manipulation and is used to introspect physical and virtual grid architectures, thanks to query expressions, and notably to modify VM placement on machines. VMScript interacts with Entropy and can be used to define administrative placement rules.

In 2011, Entropy has been integrated into a product of a newly founded start-up EasyVirt (see Sec. 7.1). Entropy has also been tested or used by our partners Orange Labs, DGFiP (direction Générale des Finances Publiques), Bull, MACIF, Logica.

Entropy is available under the LGPL license at http://entropy.gforge.inria.fr/.

5.4. FPath and FScript

Participants: Thomas Ledoux [correspondent], Frederico Alvares.

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 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 [22] 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.

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 [22] 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/s 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. Aspects

Participants: Rémi Douence, Abdelhakim Hannousse, Ismael Mejía, Jacques Noyé, Angel Núñez, Nicolas Tabareau, Mario Südholt.

We have provided results on three subject matters in the domain of aspect-oriented software development: the foundations of aspects, aspect languages and their applications, as well as the use of aspects for the manipulation of distributed systems.

As a side note on the form of this document, the reader should be aware that much of our work reported in this section is not exclusive to AOSD but also contributes to software composition issues in a larger sense. This applies, in particular, to the results cited in the subsections on aspect languages and distributed aspects.

6.1.1. Foundations of Aspects

In the domain of foundations of AOSD, we have mainly provided new results on the preservation of formal correctness properties in the presence of aspects and how AOP can be modeled using category theory.

- **Property preservation in the presence of aspects.** In general aspects can arbitrarily change the semantics of the base program. We have identified categories of aspects that preserve class of properties of the base programs [17]. This approach makes it possible to prove once and for all that a category of aspects preserve a class of properties. The categories are defined with respect to the semantics of the woven program as well as with restricted aspect languages. In this latter case, languages are defined by grammars hence checking for property preservation boils down to a syntactic check for aspects. Classes of properties are defined using a subset of temporal logic both for sequential and concurrent programs. Our framework is abstract in that it does not depend on the actual programming language but only on conditions on its small step semantics.
- **AOP and category theory.** Aspect-Oriented Programming (AOP) started ten years ago with the remark that modularization of so-called crosscutting functionalities is a fundamental problem for the engineering of large-scale applications. Originating at Xerox PARC, this observation has sparked the development of a new style of programming that is gradually gaining traction. However, AOP lacks theoretical foundations to clarify new ideas showing up in its wake. We have proposed to put a bridge between AOP and the notion of 2-category to enhance the conceptual understanding of AOP [36], [46]. Starting from the connection between the λ-calculus and the theory of categories, we have provided 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, called MinAML.

  We have later taken advantage of this new categorical framework to introduce the notion of computational 2-monads for AOP [37]. 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.
- **Membranes for AOP.** AOP aims to enhance modularity and reusability in software systems offering an abstraction mechanism to deal with crosscutting concerns. However, in most aspect-oriented languages aspects have a global view of computation that actually introduces a strong coupling between advised code and aspect code, hampering modularity. Proposals that address this problem can be classified in two categories: the first one focuses on controlling aspect scoping, i.e. the visibility of join points to aspects, while the second one focuses on protecting software units from aspects. As a new approach, we have proposed programmable membranes (inspired by the work of Boudol for distributed processes) to control aspect influence over software systems as a uniform framework that unifies and extends previous approaches [49].
- **Aspects and invertible program restructurations.** When one chooses a main axis of structural decomposition of a problem, the other axes become secondary. This is known as the tyranny of the dominant decomposition and this hinders program maintenance of secondary concerns. We propose to use automatic program transformations built on top of refactorers in order to solve this tyranny issue [47]. This offers a new approach to the expression problem by always providing the right view to the programmers.

6.1.2. Aspect Languages

This year we have provided major results on the integration of programming features for objects, events and aspects, and aspect execution infrastructures. Furthermore, we have investigated the application of aspect languages to the Java security model, component-based software development and software product lines.

- **Integrating OOP, AOP and EBP**: Object-Oriented Programming (OOP) has become the de facto programming paradigm. Event-Based Programming (EBP) and Aspect-Oriented Programming (AOP) complement OOP, covering some of its deficiencies when building complex software. Today's applications combine the three paradigms. However, OOP, EBP and AOP have not yet been properly integrated. Their underlying concepts are in general provided as distinct language constructs, whereas they are not completely orthogonal. This lack of integration and orthogonality complicates the development of software as it reduces its understandability, its composability and increases the required glue code. [16] proposes an integration of OOP, EBP and AOP leading to a simple and regular programming model. This model integrates the notions of class and aspect, the notions of event and join point, and the notions of piece of advice, method and event handler. It reduces the number of language constructs while keeping expressiveness and offering additional programming options. ECaesarJ had previously been developed based on these ideas. [16] introduces a simpler variant of it, EJava, a plain extension of Java. [24] shows that these ideas can also be easily applied to Scala, leading to EScala, and focuses on the idea that the declarative events provided by the model can be seen as object-oriented events, which, unlike global typed events, obey to the basic OO principles (OO modular reasoning, encapsulation and late-binding). An efficient implementation of EScala is described, based on the idea that an event should not be produced in the absence of at least a consumer. This is equivalent to what could be programmed by hand using variants of the observer pattern, except that all the necessary scaffolding is provided by the language compiler and runtime rather than by the programmer.
- **Prototyping and composing Aspect Languages**: [11] presents CALI (Common Aspect Languages Interpreter), a framework for rapid prototyping and composition of aspect languages based on interpreters. In practice the common interpreter is actually a thin interpretative layer built on top of Java and implemented as an AspectJ aspect. This interpreter implements common aspect mechanisms and leaves holes to be defined when developing concrete languages. The approach has been validated by implementing prototypes of significant subsets of well-known general-purpose and domain-specific aspect languages (AspectJ, EAOP, COOL and a couple of other small domain-specific aspect languages) and exploring variants of AspectJ. Languages implemented with CALI, for instance AspectJ and COOL, can then be easily composed.
- **Application to security**: It is inevitable that some concerns crosscut a sizeable application, resulting in code scattering and tangling. This issue is particularly severe for security-related concerns: it is difficult to be confident about the security of an application when the implementation of its security-related concerns is scattered all over the code and tangled with other concerns, making global reasoning about security precarious. In [21], we consider the case of access control in Java, which turns out to be a crosscutting concern with a non-modular implementation based on runtime stack inspection. We describe the process of modularizing access control in Java by means of AOP. We first show a solution based on AspectJ that must rely on a separate automata infrastructure. We then put forward a novel solution via dynamic deployment of aspects and scoping strategies. Both solutions, apart from providing a modular specification of access control, make it possible to easily express other useful policies such as the Chinese wall policy. However, relying on expressive scope control results in a compact implementation, which, at the same time, permits the straightforward expression of even more interesting policies.
- **Aspects and software components**: The relationship of aspects and components, as well as their integration as part of real-world infrastructures and application is still subject to many open issues.

  We have studied, in particular, how crosscutting concerns naturally appear when several architectural views must be considered at the same time. In this context, we have proposed a domain-specific

language to specify these architectures. We have proposed an implementation of composable controllers in Fractal as well as composition operators that makes it possible to solve aspects interferences [25]. We have also shown how to formally model the implementation in Uppaal in order to statically check aspects interference [26]. This work has principally been undertaken in the context of the PhD thesis of Abdelhakim Hannouse [12] that has been defended in Sep. 2011.

Another fundamental issue that relates components and aspects is the question whether behavioral protocols can be used to concisely define crosscutting concerns of component-based systems. Furthermore, the protocols should then be instrumental to enable (automatic) reasoning about composition properties of component systems. As part of her PhD thesis (defended in Oct. 2011) [14], Dong Ha Nguyen has defined a notion of aspects that allow crosscutting concerns to be expressed in terms of a class of non-regular behavioral protocols, so-called visibly-pushdown languages. She has developed, most notably, a constructive way of building correct component-based systems that respect such aspect-modified behavioral protocols. Furthermore, she has shown this year how composition properties can generally be verified using model checking techniques.

- **Aspects and software product lines**: In [41], we take a closer look at the difficulties of feature-oriented modularisation of product lines and demonstrate, using a case study in the domain of home automation, how a better modularisation can be achieved with the ECaesarJ programming language, through a type-safe and stable decomposition of a broad spectrum of software abstractions: classes, methods, events, and state machines, based on late binding and mixin composition. A nice property of this modularisation is that it directly captures, at the implementation level, the high-level decomposition in features, without requiring the user to resort on complex transformation technology (the technology is embedded in the compiler).

  Furthermore, ASCOLA members have co-edited and contributed to a comprehensive book on aspects, model-driven engineering and product lines [42] (see Sec. 6.2 for details).

6.1.3. Aspects for Distributed Systems

In the field of aspects for distributed systems, we have mainly extended the AWED model for distributed aspects (see Sec. 5.1) in order to define and implement a gray-box distributed composition model. We have also worked on a notion of distributed crosscuts that enable causality relationships to be captured using logical clocks.

- **Gray-box distributed composition.** Existing composition and coordination techniques for distributed applications typically support only interface-level (black-box) composition or arbitrary access to the implementation (gray-box or white-box composition). In [20], we have presented a structured approach to the composition of complex distributed software systems that require invasive modifications. Concretely, we have provided three contributions:
  - We have introduced a small kernel composition language for structured gray-box composition using invasive distributed patterns.
  - We have motivated that gray-box composition approaches should be defined and evaluated in terms of the flexibility and control they provide; a notion of degrees of invasiveness has been introduced to help assess this trade-off.
  - We have applied and evaluated it in the context of several studies involving two real-world software systems: benchmarking of grid algorithms with NASGrid and transactional replication with JBoss Cache. As a main result, we have shown that gray-box composition using invasive distributed patterns allows the declarative and modular definition of evolutions of real-world applications that need moderate to high degrees of invasive modifications.

  We have then provided a first framework-based implementation based on our AWED tool and applied it to evolution tasks in OpenMRS, a health information system [31]. The composition framework supports the definition of expressive pattern-based invasive compositions based on the skeleton paradigm for distributed patterns. Concretely, we have shown that the composition framework

enables the concise definition of an evolution scenario of OpenMRS that supports the consolidation of patient data from different distributed instances.

Finally, we have studied a model that integrates distributed aspects and actors and thus improves on the robustness properties of existing models for distributed aspects [32].

- **Causal aspects for distributed applications.** In [45], we have applied logical clocks à la Lamport in order to define causality between distributed join points. This enables us to declaratively define pointcuts in a distributed context such as web-based applications in JavaScript. The definition of advice can then be simplified, because they are executed only when necessary and do not have to maintain and check crosscutting information.

6.2. Software Composition

Participants: Jean-Claude Royer, Hervé Grall, Christine Louberry, Mario Südholt.

ASCOLA's work on software composition addresses the foundations of software composition methods, the definition and implementation of concrete composition techniques and their application to different functionalities and application domains. This year we have contributed to the questions of how to identify components in legacy software, how to effectively deploy and reconfigure component-based software in pervasive environments, and how to apply software composition techniques to Cloud security as well as software product lines.

- **Component identification.** The communication integrity property is one of the major principles to implement software architectures, however, there is a lack of tooling for assessing the quality of components codes. To cope with this issue, we defined, in [23], a Java component model and a tool for identifying component types based on the communication integrity property. We apply it to several case studies and compare the result with the SOMOX component recovery tool.
- **QoS-driven deployment and reconfiguration of pervasive applications.** [28] presents the Kalimucho platform, a platform for the dynamic reconfiguration of applications on mobiles and constraints devices. First this article focuses on the heuristics implemented by Kalimucho. They support finding a configuration and a deployment matching two criteria: utility and durability. Moreover, we present a case study to experiment the approach of Kalimucho. It confirms that Kalimucho provides a satisfactory execution time for mobile devices.

  A second result we have contributed in this context is a two-dimensional QoS model for Kalimucho that supports the QoS-driven deployment of mobile applications [29]. After presenting the definitions of the context and the quality of service considered in this work, this article describes the QoS model and the process allowing finding the best configuration and deployment. Finally, we present several results concerning the execution time of the deployment process of Kalimucho with different devices.
- **Securing the Cloud: cross domain and multi-level concerns.** The evolution of new deployment architectures, as illustrated by the move towards mobile platforms and the Internet Of Services, and the introduction of new security regulations (imposed by national and international regulatory bodies, such as SOX4 or BASEL5) are important constraints in the design and development of service composition. In such a context, it is not sufficient to apply the corresponding adaptations only at the service orchestration or at the choreography level; there is also the need for controlling the impact of new security requirements on several architectural layers. In [35] we have presented a new service model that supports the clean modularization of such crosscutting security concerns.
- **Software composition and product lines.** As part of the AMPLE project (see http://www.ample-project.net) we have co-edited a book [42] covering the main scientific solutions and techniques on new methods and techniques for software product lines that have been developed in the project. This project aims at improving traditional software product lines engineering using advanced software engineering namely model-driven and aspect-oriented engineering approaches. Software composition takes an important part in this book and it appears in several chapters with models, events, aspects and components.

Chapter [43] provides an overview of software product lines, model-driven engineering and aspect-oriented software development. The challenges to address and the expected benefits are drawn, this is concluded by an overview of the AMPLE approach and its tool chain.

Chapter [40] reviews the specificities of traceability in a product line context starting by identifying the challenges of maintaining traceability for traditional system development and for software product lines. This work defines the concepts that should guide the adoption of a traceability environment for product lines and illustrates these specifications with a concrete example of a traceability repository. It also provides examples of scenarios that use this traceability environment to solve concrete problems.

6.3. Cloud Computing, Virtualization and Data Centers

Participants: Frederico Alvares, Gustavo Bervian Brand, Thomas Chavrier, Fabien Hermenier, Adrien Lèbre, Thomas Ledoux, Guillaume Le Louët, Jean-Marc Menaud, Hien Nguyen Van, Rémy Pottier, Flavien Quesnel.

In the context of Cloud computing ASCOLA members have principally worked this year on capacity planning solutions for large scale distributed system. Capacity planning is the process of planning, analyzing, sizing, managing and optimizing capacity to satisfy demand in a timely manner and at a reasonable cost.

Applied to distributed systems like the Cloud, a capacity planning solution must mainly provide necessary resources for the proper execution of applications and respect service-level agreements in a large distributed environment.

The main challenges in this context are: scalability, fault tolerance and reactivity of the solution in a large-scale distributed system; analyzing, sizing, and optimizing resources to minimize the cost (energy, human, hardware etc.); and profiling, adapting application to ensure the quality of service (throughput, response time, availability etc.).

Our solutions are mainly based on virtualized infrastructures. Our main results concern the management and the execution of applications by leveraging virtualization capabilities on cloud infrastructures and the investigation of solutions that aim at optimizing the trade-off between performance and energy costs of both applications and Cloud resources.

6.3.1. Virtualization and Job Management

This year, in cooperation with the Myriads project-team from INRIA Rennes-Bretagne Atlantique, we have continued to address resource-management issues concerning the federation of very large scale platforms. We have completed our approach aiming at the automatic adaptation of both hardware and software resources to the needs of the applications through a unique method. For each application, scientists describe the requirements in terms of both hardware and software expectations through the definition of a *Virtual Platform (VP)* and a *Virtual System Environment (VSE)* [18]

In addition, we started to address the design and the implementation of a fully distributed cloud OS. Designing and implementing such models is a tedious and complex task. However, as well as research studies on OSes and hypervisors are complementary at the node level, we showed that virtualization frameworks can benefit from lessons learned from distributed operating system proposals [34]. Leveraging this preliminary result, we designed and developed a first proof-of-concept of a fully distributed scheduler [33]. This system makes it possible to schedule VMs cooperatively and dynamically in large scale distributed systems. Preliminary results showed that our scheduler was more reactive. This building block is a first element of a more complete cloud OS, entitled DISCOVERY (DIStributed and COoperative mechanisms to manage Virtual EnviRonments autonomicallY) [30]. The ultimate goal of this system is to overcome the main limitations of the traditional server-centric solutions. The system, currently under development, relies on a peer-to-peer model where each agent can efficiently deploy, dynamically schedule and periodically checkpoint the virtual environments s/he manage.

We have contributed new results on the Entropy software and extended our solution VM that features dynamic consolidation. In [44] and [38] we extended our dynamic consolidation manager to take into account not only resource constraints but also the placement constraints of highly available (HA) applications. In fact, most previous dynamic consolidation systems optimize the placement of the VMs according to their resource usage but do not consider the application placement constraints that are required to achieve both high availability and scalability. Our approach provides efficient dynamic consolidation while guaranteeing to the application administrator that placement requirements will be satisfied and relieving the data center administrator of the burden of considering the constraints of the applications when performing maintenance.

In the same domain, Jean-Marc Menaud has defended his habilitation (HdR - *Habilitation à diriger des recherches*) [13] on Jun. 22, 2011. One part of this HDR focuses on dynamic adaptation strategies for cluster administration. We have proposed a dedicated language for virtual machines management and one particular feature of our solution is to use a constraint solver to provide an appropriate placement. Based on these results, our recent contributions address the problem of data center energy consumption.

Moreover, we have continued to analyze how energy concerns can be addressed in large scale distributed infrastructures.

6.3.2. Optimization of Energy Consumption in Data Centers

As a direct consequence of the increasing popularity of Cloud Computing solutions, data centers are growing at a fast rate and hence have to face difficult energy consumption issue. Available solutions rely on Cloud Computing models and virtualization techniques to scale up/down applications 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 QoS properties of applications and their energy footprint. In [22], we propose a self-adaptation approach that considers both application internals and system properties to reduce the energy footprint in cloud infrastructures. Each application and the infrastructure are equipped with their own control loop, which allows them to autonomously optimize their executions. Simulations show that the approach may lead to appreciable energy savings without interfering on application provider revenues.

6.3.3. Cloud Computing and SLA Management

Cloud computing is a paradigm for enabling remote, on-demand access to a set of configurable computing resources as a service. The pay-per-use model enables service providers to offer their services to customers at different Quality-of-Service (QoS) levels. These QoS parameters are used to compose service-level agreements (SLAs) between a service provider and a service consumer. A main challenge for a service provider is to manage SLAs for its service consumers, i.e., automatically determine the appropriate resources required from the lower layer in order to respect the QoS requirements of the consumers. In [27], we have proposed an optimization framework driven by consumer preferences to address the SLA dependencies problem across the different cloud layers as well as the need of flexibility and dynamicity required by the domain of Cloud computing. Our approach aims at selecting the optimal vertical business process designed using cross-layer cloud services and enforcing SLA dependencies between layers. Experimental results demonstrate the flexibility and effectiveness of our approach.

6.4. Foundations of program semantics

Participants: Nicolas Tabareau, Guilhem Jaber.

ASCOLA team members have contributed several results to the foundations of program semantics.

6.4.1. Program Equivalences

Reasoning about program equivalence is a major problem in semantics. This very old topic has many applications, e.g., program verification, compiler correctness or representation independence. It has been understood since the late 1960s that tools and structures arising in mathematical logic and proof theory can usefully be applied to the development of reasoning principles for program equivalence. In recent years, based

on the seminal work of Pitts and Stark, the notion of logical relation appeared to be very fruitful for proving the equivalence of programs in the presence of features like general recursive types, general (higher-order) mutable references, and first-class continuations. We have studied the notion of logical relations for proving program equivalences of low level machine codes [39].

We have also developed a forcing theory inspired by Cohen's forcing to increase the power of a semantics model just as the latter makes it possible to enrich a set theoretical universe. In this way, we can define a new generation of logical relations—that can be introduced modularly using forcing theory—to be used for proving program equivalence for low level languages, concurrent languages or domain specific languages. [48]

7. Contracts and Grants with Industry

7.1. Start-up EasyVirt

EasyVirt is a start-up, founded in Jul. 2011, that specializes in the management and optimization of data centers. It provides a software solution, named **btrCloud**, that has been developed by the Ascola research team. This software supports data center real-time monitoring, the reduction of their energy footprint and security hardening of data center infrastructures. Part of EasyVirt's product is based, in particular, on Ascola's Entropy virtual machine manager, see Sec. 5.3.

7.2. Contracts with Industry

PhD about Virtualisation in data centers (Cifre Hien Van Nguyen with Orange Labs)

To satisfy QoS properties of clients of data centers (such as the expected request rates), a standard data center statically allocates resources according to the worst-case conditions defined in the contract formally negotiated by both parties. As a result, the data center must be oversized and is most of the time underused. From the point of view of the hosting provider (who hosts multiple client applications), the problem is to define an optimal resource allocation, which maximizes client criteria but minimizes the costs of the provider.

By using our current results around Entropy, Hien Nguyen Van's PhD work defines relations between QoS rules and resources needs (CPU, memory) by designing a specific domain-specific language for managing data centers, in particular their energy consumption.

This work was supported by Orange Lab for an amount of 27 KEUR.

8. Partnerships and Cooperations

8.1. National Initiatives

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

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

The project CESSA is an (industrial) ANR project running for 36 months. It was accepted in Jun. 2009 for funding amounting to 290 KEUR for ASCOLA from Dec. 2009 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 two surveys on models for service-oriented architectures and security properties. Furthermore, they have set up a blog for SAP's worldwide developper community.

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

8.1.2. Cool-IT (FUI)

Participant: Jean-Marc Menaud.

The Cool-IT project is an (industrial) FUI project running for 24 months. It was accepted in Sept. 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 electrical consumption of data centers.

To this end, the COOL IT project develops processes for cooling computer servers, optimizes the servers power chain supply, implements tools and methods for collecting energy data in real time, and specifies methods for controlling the data centers consumption as a tradeoff between the computational power needed, its availability, and its energy consumption.

8.1.3. Entropy (ANR/Emergence)

Participant: Jean-Marc Menaud.

The Entropy project is an (industrial) ANR/Emergence project running for 18 months. It was accepted in Dec. 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 an industrial business based 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.

8.1.4. 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 is a paradigm for enabling remote, on-demand access to a set of configurable computing resources. However, the ad-hoc management of a cloud in terms of Quality of Service (QoS) and Service Level Agreement (SLA) poses significant challenges to the performance, availability, energy consumption and economical costs of the cloud.

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 the global architecture and framework for the SLAaaS model and has provided a solution for self-optimisation of the energy footprint in cloud infrastructures [22].

8.1.5. SelfXL (ANR/ARPEGE)

Participant: Jean-Marc Menaud.

The SelfXL project is an (industrial) ANR/ARPEGE project running for 36 months. It was accepted in Jul. 2008 for funding amounting to 315 KEUR (ASCOLA only) from Jan. 2009 on.

The SelfXL project aims at investigating abstractions and implementation techniques (language mechanisms, runtime structures...) for complex and large-scale autonomic systems. The scope of this project encompasses any system that has a high software complexity (distributed, size of code etc.) and is large-scale in terms of size and heterogeneity of resources and software. Systems to be targeted range from cluster computing to embedded systems, including legacy software.

Two main issues will be addressed by SelfXL: How to implement administration policies for complex system and how to coordinate administration policies in a complex system? Regarding the first issue, SelfXL proposes to explore the DSL programming approach, i.e., designing specific languages for defining specific kinds of administration policies (self-repair, self-optimization, self-protection). The general use of DSLs would ensure the correctness of the policies.

We propose to design a decision module based on Constraints Programming (CP). As the Rules Based Systems (RBS) or the Event Condition Action (ECA) approach, CP belongs to the declarative paradigm but does not share the major drawback of the other approaches when some rules are simultaneously asserted. This is the case when there is an overlap between the domain or the target of rules.

Finally, we propose to extend the Jasmine autonomic administration platform (http://wiki.jasmine.objectweb.org) for supporting a decentralized and hierarchical infrastructure to address the large-scale administration.

8.2. European Initiatives

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

Participant: Adrien Lèbre.

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. COST IC0804

Participant: Jean-Marc Menaud.

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: University of Chile (Chile)

  Laboratory: Computer Science Department
- 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.

8.3.2. Visits of International Scientists

8.3.2.1. Invited Researchers and Professors

- Prof. Awais Rashid (from Feb. until Mar. 2011)

  Subject: Sustainable Software for a Sustainable World

  Institution: Lancaster University, U.K.

  Pays de la Loire regional chair in Computer Science at École des Mines de Nantes, 2009-2011
- Dr. Paolo Anneda (Nov. 2011)

  Subject: Optimization of the energy footprint of Cloud infrastructures

  Institution: CRS4, Italie

  LINA (CNRS) grant

8.3.2.2. Internships

- Mohammad Mohammad Atiqul Haque (from Mar. until Jul. 2011)

  Subject: Impact of dynamic VM scheduling in cloud platforms

  Institution: Colorado State University (United States)

  INRIA grant
- Mauricio De Diana (from Mar. until Jul. 2011)

  Subject: Federation of distributed file systems for grid and cloud architectures

  Institution: Universidade de São Paulo (Brazil)

  INRIA grant

9. Dissemination

9.1. Animation of the Community

9.1.1. Animation

9.1.1.1. International

Aspect-oriented Software Association (AOSA): Mario Südholt has been elected the chair of this international organization (headquarted in California) in Apr. 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.

DSAL: Jacques Noyé has co-organized the 6th international workshop on "Domain-Specific Aspect Languages" (DSAL 2011) co-located with AOSD 2011 in Porto de Galinhas, Pernambuco, Brazil.

VTDC: Adrien Lèbre has been the general chair of the 5th international workshop on "Virtualization Technologies in Distributed Computing" (VTDC 2011) co-located with HPDC 2011 in San José, CA, United-States.

Cloud computing summer school: Adrien Lèbre gave a lecture at the European summer school organized by EIT-ICT lab and the CONTRAIL project in the "Presqu'ile de Giens" Hyères-les-Palmiers, France (Jun. 27 - Jul. 1, 2011). http://contrail-project.eu/summerschool-2011.

OW2: Jean-Marc Menaud has been a member of the OW2 board committee since 2010.

9.1.1.2. France

CNRS GDR GPL: Pierre Cointe is a member of the scientific committee of the GDR GPL (*Génie de la Programmation et du Logiciel*).

CNRS GDR ASR: Jean-Marc Menaud is a member of the scientific committee of the CNRS GDR ASR (*Architectures, Systèmes, Réseaux*) and in charge of the System action.

ACM/SIGOPS: Jean-Marc Menaud was the vice-chair of the French ACM/SIGOPS Chapter (ASF) from Mar. 2008 to Mar. 2011.

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.

Energy Issues in Distributed Systems: Jean-Marc Menaud organized the workshop *Journée Thèmes Emergents* (JTE) on Energy Issues in Distributed Systems in Paris (May 30 to Jun. 1, 2011).

France Grille: Adrien Lèbre gave a lecture at the Cloud Computing days organized by France Grille in Lyon, France, Oct. 21, 2011.

SPL: Jean-Claude Royer participated in the organization of the workshop *Journée lignes de produits logiciels*, Paris, Oct. 21, 2011. http://www.jldp.org/.

9.1.1.3. PdL Region

Quartier de la création de Nantes: Pierre Cointe organized with Bertrand Stiegler (Ars Industrialis) a workshop on Web Sciences and Software Studies (May 27, 2011).

Regional Doctoral School STIM: Thomas Ledoux co-organized JDOC, an event promoting contact between PhD students of the Doctoral School (Apr. 14, 2011), and the Welcome Day event for all the new PhD students (Nov. 3, 2011).

9.1.2. Steering, Journal, Conference Committees

P. Cointe is a member of AITO and the ECOOP steering committee. He was a member of the NOTERE 2011 program committee.

A. Lèbre was a member of the program committee of the 11th IEEE International Conference on Scalable Computing and Communications (ScalCom 2011), the IEEE International Conference on Cloud and Green Computing (CGC 2011), the 20th EUROMICRO Conference on Parallel, Distributed and Network-based and Processing (special session on Cloud Computing), the 5th Workshop on System-level Virtualization for High Performance Computing (HPCVirt 2011, co-located with the EUROPAR conference), and the 4th IEEE Worshop on Real-Time Service Oriented Architecture and Applications (RTSOAA 2011, co-located with SOCA 2011).

T. Ledoux was a member of program committees for the following conferences: 2nd International Workshop on Green Computing Middleware (GCM'11), 10th International Workshop on Adaptive and Reflective Middleware (ARM'11), 1st Workshop on Middleware and Architectures for Autonomic and Sustainable Computing (MAASC'11), International Conference on Green Computing ICGREEN 2011, International Conference on ICT as Key Technology against Global Warming (ICT- GLOW'11). He was also a member of the program committee for the book "Achieving Federated and Self-Manageable Cloud Infrastructures: Theory and Practice" (IGI Global, 2012).

J.-M. Menaud is a member of the (RenPar/CFSE/Sympa) steering committees. He has served on the program committee of the IEEE/ACM Inter. Conference on Green Computing and Communications (GreenCom2011), the first International Workshop on Autonomic Systems enabling Green Computing (ASGC 2011) as part of The Tenth International Conference on Networks (ICN 2011), the first International Conference on Smart Grids, Green Communications and IT Energy-aware Technologies (ENERGY 2011), the first International Conference on ICT as Key Technology for the Fight against Global Warming (IT-GloW '11), the 6th Workshop on Virtualization in High-Performance Cluster and Grid Computing (VHPC '11) as part of Euro-Par 2011 and the 5th Workshop on System-level Virtualization for High Performance Computing (HPCVirt 2011) as part of EuroSys 2011.

J-C. Royer is a member of the editorial board of the journal TSI. He was a member of the program committee of CAL 2011, CNSI 2011 and SSNE 2011.

M. Südholt 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.

N. Tabareau was a member of the program committee of the workshop LOLA (Syntax and Semantics of Low-Level Languages) 2011, satellite of LICS 2011.

9.1.3. Thesis Committees

P. Cointe was a reviewer of Julien Mercadel's PhD (University of Bordeaux, Oct. 10).

T. Ledoux has taken part in two PhD evaluation committees: Roméo Said (UBS, 23 Feb.), Loris Bouzonnet (LIG-IMAG, 16 Sep.)

A. Lèbre was member of two PhD committees: Jérôme Gallard (Rennes, 6 May) and Alexandra CARPEN AMARIE (Rennes, 15 Dec.).

J.-M. Menaud was reviewer of one PhD: Alain-B. TCHANA (Oct. 29, 2011) in Toulouse.

J-C. Royer was reviewer of three PhDs: Mario Sanchez Puccini in Bogota, Anthony Hock-Koon in Nantes, Carlos Parra in Lille and member of Angel Núñez's PhD committee.

M. Südholt has served as a reporting jury member for the PhD of Slim Kallel (TU Darmstadt, Germany, and ENIS, Sfax, Tunisia) in May 2011.

9.1.4. Evaluation Committees and Expertise

P. Cointe was a member of the AERES visiting committee for MIPS (Mulhouse, Nov. 15-16, 2011). He served in four selection/hiring committees (EMN, Polytech'Nantes and University of Nantes) for associate and full professor positions.

J.-M. Menaud acted as an expert for the ANR International White Program 2011 call.

J-C. Royer acted as an expert for the ANR blanc SIMI 2 2011 call.

M. Südholt has served as an expert to the Cofecub French-Brazilian exchange program.

N. Tabareau was a member of the selecting committee for the position of assistant professor (MC298) at École Normale Supérieure (ENS) de Cachan, Ker Lann and assistant professor (MC04867) at Laboratoire Informatique de Nantes Atlantique (LINA).

9.2. Collective Duties

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.

T. Ledoux: 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. He is a member of the administrative committee of the INRIA Rennes-Bretagne Atlantique. Finally, he is the Apprenticeship program manager in Software engineering at Ecole des Mines de Nantes.

J.-M. Menaud is the deputy of the computer science department at Ecole des Mines de Nantes, in charge of the scholarship program.

M. Südholt: 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.

A. Lèbre is involved in the direction committee of the Grid'5000/Aladdin platform. In addition, He is in charge of the Virtualization working group since 2008.

9.3. Teaching

9.3.1. Steering and Main Lecturing Activities

Members of the team has been responsible for or contributing to the following main study programs:

- The team is a main contributor (of approx. 1000 hours in 2010-2011) to the informatics education (2600 hours in total) within the four-year engineering program at École des Mines de Nantes.

  The team is steering, chairing and the main contributor to a fourth-year informatics specialization for a total teaching volume of 800 hours of which Ascola is contributing 210 hours.
- The team has designed and set up a three-year Apprenticeship engineering program for a total teaching volume of 1800 hours (816 in informatics). The first year of the formation started in Oct. 2011; Ascola is contributing 95 hours in 2011-2012.
- The team is steering and teaching a 48-hour module on AOSD in an MSc program that is organized jointly by University of Nantes and École des Mines de Nantes.
- Members of the team have taught different courses at different study levels in Rennes that are mainly organized by University of Rennes and the research institutes Irisa and Inria.

9.3.2. Habilitations (HdR) and PhD Supervision

Jean-Marc Menaud has defended his habilitation (HDR, *Habilitation à Diriger des Recherches*) on Jun. 22, 2011 [13].

This year 5 PhD theses have been defended by students of the ASCOLA team:

- Hien Nguyen Van [15]
- Angel Núñez [16]
- Abdelhakim Hannousse [12]
- Dong Ha Nguyen [14]
- Ali Assaf [11]

10. Bibliography

Major publications by the team in recent years

[1] F. BALIGAND, N. RIVIERRE, T. LEDOUX. *QoS Policies for Business Processes in Service Oriented Architectures*, in "Proc. of the 6th Int. Conference on Service Oriented Computing (ICSOC)", Sydney, Australia, Springer-Verlag, December 2008, vol. 5364, p. 483–497.

[2] L. D. BENAVIDES NAVARRO, R. DOUENCE, M. SÜDHOLT. *Debugging and testing middleware with aspect-based control-flow and causal patterns*, in "Proc. of the ACM/IFIP/USENIX 9th Int. Middleware Conference", Leuven, Belgium, Lecture Notes in Computer Science, Springer-Verlag, December 2008, vol. 5346, p. 183-202.

[3] B. DE FRAINE, E. ERNST, M. SÜDHOLT. *Essential AOP: The A Calculus*, in "European Conference on Object-Oriented Programming (ECOOP'10)", Springer Verlag, June 2010.

[4] B. DE FRAINE, M. SÜDHOLT, V. JONCKERS. *StrongAspectJ: Flexible and Safe Pointcut/Advice Bindings*, in "Proc. of the 7th ACM Int. Conf. on Aspect-Oriented Software Development (AOSD'08)", M. MEZINI (editor), ACM Press, March 2008, p. 60–71, Distinguished Paper Award.

[5] 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.

[6] 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, SPRINGER-VERLAG (editor), Lecture Notes in Computer Science, June 2010, p. 74-92.

[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 (editor), 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

Doctoral Dissertations and Habilitation Theses

[11] A. ASSAF. *A Common Aspect Languages Interpreter*, École des Mines de Nantes, Université de Nantes, October 2011.

[12] A. HANNOUSSE. *Aspectualizing Component Models: Implementation and Interferences Analysis*, École des Mines de Nantes, Université de Nantes, November 2011, http://tel.archives-ouvertes.fr/tel-00657285.

[13] J.-M. MENAUD. *Adaptation dynamique et transparente de systèmes patrimoniaux complexes*, Université de Nantes, June 2011, HDR.

[14] D. H. NGUYEN. *A VPA-based Aspect Language*, École des Mines de Nantes, Université de Nantes, October 2011, http://tel.archives-ouvertes.fr/tel-00642636.

[15] H. NGUYEN VAN. *Gestion de ressources virtualisées pour plates-formes d'hébergement de services*, École des Mines de Nantes, Université de Nantes, February 2011.

[16] A. NÚÑEZ. *A Programming Model Integrating Classes, Events and Aspects*, École des Mines de Nantes, Université de Nantes, June 2011, http://tel.archives-ouvertes.fr/tel-00656649.

Articles in International Peer-Reviewed Journal

[17] S. DJOKO DJOKO, R. DOUENCE, P. FRADET. *Aspects Preserving Properties*, in "Science of Computer Programming", October 2011 [*DOI :* 10.1016/J.SCICO.2011.10.010], http://hal.inria.fr/inria-00638852/en.

[18] J. GALLARD, A. LÈBRE, C. MORIN, T. NAUGHTON, S. SCOTT, G. VALLÉE. *Architecture for the Next Generation System Management Tools*, in "Future Generation Computer Systems", 2011 [*DOI :* 10.1016/J.FUTURE.2011.06.003], http://hal.inria.fr/inria-00605143/en.

[19] C. LOUBERRY, P. ROOSE, M. DALMAU. *Kalimucho: Contextual Deployment for QoS Management*, in "Springer LNCS", June 2011, vol. 6723, p. 43-56, http://hal.inria.fr/hal-00607375/en.

[20] I. MEJIA, M. SÜDHOLT. *Structured and flexible gray-box composition using invasive distributed patterns*, in "International Journal on Computer Science and Information Systems", March 2011, vol. 6, 13, ISBN = ISSN: 1646-3692, http://hal.inria.fr/inria-00583673/en.

[21] R. TOLEDO, A. NÚÑEZ, É. TANTER, J. NOYÉ. *Aspectizing Java Access Control*, in "IEEE Transactions on Software Engineering", January 2011 [*DOI :* 10.1109/TSE.2011.6], http://hal.inria.fr/inria-00567489/en.

International Conferences with Proceedings

[22] F. ALVARES DE OLIVEIRA JR., T. LEDOUX. *Self-management of applications QoS for energy optimization in datacenters*, in "2nd International Workshop on Green Computing Middleware (GCM'2011)", Portugal, 2011, p. 0-0, http://hal.inria.fr/hal-00641902/en.

[23] H. ARBOLEDA, J.-C. ROYER. *Component types qualification in Java legacy code driven by communication integrity rules*, in "ISEC 2011 : India Software Engineering Conference", Thiruvananthapuram, Kerala, India, ACM, February 2011, p. 155–164 [*DOI :* 10.1145/1953355.1953377], http://hal.inria.fr/hal-00621001/en.

[24] V. GASIŪNAS, L. SATABIN, M. MEZINI, A. NÚÑEZ, J. NOYÉ. *EScala: Modular Event-Driven Object Interactions in Scala*, in "10th International Conference on Aspect-Oriented Software Development (AOSD 2011)", Porto de Galinhas, Brazil, ACM Press, March 2011 [*DOI :* 10.1145/1960275.1960303], http://hal.inria.fr/inria-00559183/en.

[25] A. HANNOUSSE, R. DOUENCE, G. ARDOUREL. *Composable Controllers in Fractal: Implementation and Interference Analysis*, in "the 37th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA'11)", Oulu, Finland, September 2011, 99, http://hal.inria.fr/hal-00606269/en.

[26] A. HANNOUSSE, R. DOUENCE, G. ARDOUREL. *Static Analysis of Aspect Interaction and Composition in Component Models*, in "the 10th International Conference on Generative Programming and Component Engineering (GPCE'11)", Portland, Oregon, United States, October 2011, 18, http://hal.inria.fr/hal-00606270/en.

[27] Y. KOUKI, T. LEDOUX, R. SHARROCK. *Cross-layer SLA selection for Cloud services*, in "2011 First International Symposium on Network Cloud Computing and Applications", Toulouse, France, November 2011, p. 143-147, http://hal.inria.fr/hal-00642532/en.

[28] C. LOUBERRY, P. ROOSE, M. DALMAU. *Kalimucho : Plateforme d'Adaptation des Applications Mobiles*, in "Actes Conférences NOTERE", Paris, France, May 2011, p. 83-90, http://hal.archives-ouvertes.fr/hal-00593459/en/.

[29] C. LOUBERRY, P. ROOSE, M. DALMAU. *Kalimucho: Contextual Deployment for QoS Management*, in "Distributed Applications and Interoperable Systems", Reykjavik, Iceland, Lecture Notes in Computer Science, Springer, June 2011, vol. 6723, p. pp.43-56 [*DOI :* 10.1007/978-3-642-21387-8], http://hal.inria.fr/hal-00596372/en.

[30] 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.

[31] I. MEJIA, M. SÜDHOLT, L. D. BENAVIDES NAVARRO. *Invasive composition for the evolution of a health information system*, in "2nd International Workshop on Variability & Composition (VariComp 2011)", Pernambuco, Brazil, ACM DIGITAL LIBRARY (editor), ISBN: 978-1-4503-0646-1, ACM New York, NY, USA, March 2011, ACM 978-1-4503-0646-1/11/03 [*DOI :* 10.1145/1961359.1961362], http://hal.inria.fr/inria-00567598/en.

[32] I. MEJIA, M. SÜDHOLT. *Towards a robust model for distributed aspects*, in "1st Workshop In Modularity in Systems Software (MISS)", Pernambuco, Brazil, ACM DIGITAL LIBRARY (editor), ISBN: 978-1-4503-

0647-8, ACM New York, NY, USA, March 2011 [*DOI :* 10.1145/1960518.1960523], http://hal.inria.fr/inria-00567604/en.

[33] F. QUESNEL, A. LÈBRE. *Cooperative Dynamic Scheduling of Virtual Machines in Distributed Systems*, in "6th Workshop on Virtualization in High-Performance Cloud Computing", Bordeaux, France, August 2011, XX, http://hal.inria.fr/hal-00622862/en.

[34] F. QUESNEL, A. LÈBRE. *Operating Systems and Virtualization Frameworks: From Local to Distributed Similarities*, in "19th Euromicro International Conference on Parallel, Distributed and Network-Based Computing", Ayia Napa, Cyprus, February 2011, 495, http://hal.inria.fr/hal-00538948/en.

[35] M. SABIR IDREES, G. SERME, Y. ROUDIER, A. SANTANA DE OLIVEIRA, H. GRALL, M. SÜDHOLT. *Evolving Security Requirements in Multi-Layered Service-Oriented-Architectures*, in "4th International Workshop on Autonomous and Spontaneous Security", Leuven, Belgium, September 2011, http://hal.inria.fr/inria-00614163/en.

[36] N. TABAREAU. *Aspect oriented programming: a language for 2-categories*, in "Proceedings of the 10th international workshop on Foundations of aspect-oriented languages", Porto de Galinhas, Brazil, ACM, 2011, p. 13–17 [*DOI :* 10.1145/1960510.1960514], http://hal.inria.fr/inria-00583429/en.

[37] 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/en.

National Conferences with Proceeding

[38] F. HERMENIER, J. LAWALL, G. MULLER, J.-M. MENAUD. *Consolidation dynamique d'applications Web haute-disponibilité*, in "Actes Conférences CFSE", Rennes, France, May 2011.

Conferences without Proceedings

[39] G. JABER, N. TABAREAU. *The Journey of Biorthogonal Logical Relations to the Realm of Assembly Code*, in "Workshop LOLA 2011, Syntax and Semantics of Low Level Languages", Toronto, Canada, June 2011, http://hal.inria.fr/hal-00594386/en.

Scientific Books (or Scientific Book chapters)

[40] N. ANQUETIL, U. KULESZA, R. MATEUS, R. MITSCHKE, A. MOREIRA, J.-C. ROYER, A. RUMMLER. *Managing Information Flow in SPL Development Processes*, in "Aspect-Oriented, Model-Driven Software Product Lines, The AMPLE way", Cambridge University Press, September 2011, p. 222–262, ISBN 978-0-521-76722-4, http://hal.inria.fr/hal-00620995/en.

[41] V. GASIŪNAS, A. NÚÑEZ, J. NOYÉ, M. MEZINI. *Product line implementation with ECaesarJ*, in "Aspect-Oriented, Model-Driven Software Product Lines - The AMPLE Way", A. RASHID, J.-C. ROYER, A. RUMMLER (editors), Cambridge University Press, September 2011, http://hal.inria.fr/inria-00607804/en.

[42] A. RASHID, J.-C. ROYER, A. RUMMLER. *Aspect-Oriented, Model-Driven Software Product Lines The AMPLE Way*, Cambridge University Press, September 2011, ISBN: 9780521767224, http://hal.inria.fr/hal-00620981/en.

[43] A. RASHID, J.-C. ROYER, A. RUMMLER. *Introduction*, in "Aspect-Oriented, Model-Driven Software Product Lines, The AMPLE way", Cambridge University Press, September 2011, p. 3–26, ISBN 978-0-521-76722-4, http://hal.inria.fr/hal-00620988/en.

Research Reports

[44] F. HERMENIER, J. LAWALL, J.-M. MENAUD, G. MULLER. *Dynamic Consolidation of Highly Available Web Applications*, INRIA, February 2011, n^{o} RR-7545, http://hal.inria.fr/inria-00567102/en.

[45] P. LEGER, É. TANTER, R. DOUENCE. *Modular and Flexible Causality Control on the Web*, INRIA, June 2011, n^{o} RR-7742, http://hal.inria.fr/inria-00626363/en.

[46] N. TABAREAU. *Aspect Oriented Programming: a language for 2-categories*, INRIA, February 2011, n^{o} RR-7527, http://hal.inria.fr/inria-00470400/en.

Other Publications

[47] J. COHEN, R. DOUENCE. *Views, Program Transformations, and the Evolutivity Problem in a Functional Language*, 2011, 19 pages, http://hal.inria.fr/hal-00481941/en.

[48] G. JABER, N. TABAREAU. *Decomposing Logical Relations with Forcing*, 2011, http://hal.inria.fr/hal-00585717/en.

[49] É. TANTER, N. TABAREAU, R. DOUENCE. *Exploring Membranes for Controlling Aspects*, 2011, http://hal.inria.fr/inria-00592133/en.

References in notes

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

[51] 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.

[52] 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.

[53] 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.

[54] 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.

[55] 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.

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

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

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

[59] 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.

[60] 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.

[61] 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.

[62] 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.

[63] 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, ftp://ftp.inria.fr/INRIA/publication/publi-pdf/RR/RR-4435.pdf.

[64] 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.

[65] 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.

[66] 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.

[67] 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.

[68] 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.

[69] 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.

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

[71] 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.

[72] 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, AspectJ web site: http://aspectj.org, http://www.eclipse.org/aspectj/.

[73] 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.

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

[75] 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.

[76] 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.

[77] 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.

[78] 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.

[79] 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.

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

[81] 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.

[82] 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.

[83] 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.

[84] 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.

[85] 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.

[86] 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.

[87] 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.

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

[89] 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.

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

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

[92] 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.

[93] 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.

[94] 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.

[95] 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.

[96] 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.