INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team Ascola

Aspect and composition languages

Rennes - Bretagne-Atlantique

Theme : Distributed Systems and Services

Table of contents

1. Team

Research Scientists

Rémi Douence [Junior Researcher INRIA on leave from EMNantes since 1st Sep.]
Thomas Ledoux [Junior Researcher INRIA on leave from EMNantes until 31 Aug.]
Nicolas Tabareau [Junior Researcher INRIA]

Faculty Members

Pierre Cointe [Professor, EMNantes, HdR]
Hervé Grall [Associate Professor, EMNantes]
Fabien Hermenier [Associate Professor (temporary), until Oct. 10]
Adrien Lèbre [Associate Professor, EMNantes]
Jean-Marc Menaud [Associate Professor, EMNantes]
Jacques Noyé [Vice team leader: Associate Professor, EMNantes]
Jean-Claude Royer [Professor, EMNantes, HdR]
Mario Südholt [Team leader: Associate Professor, EMNantes, HdR]

Technical Staff

Thomas Chavrier [INRIA (ADT VASP), since Oct. 10]
Mahmoud Ben Hassine [INRIA (together with TRISKELL on project GALAXY), until Oct. 10]

PhD Students

Akram Ajouli [MINES grant, since Oct. 10]
Diana Allam [MINES (ANR CESSA) grant, since Oct. 10]
Frederico Alvares [MESR grant, since Oct. 09]
Gustavo Bervian Brand [MCITN SCALUS grant, since Sept. 10]
Kelly Garcés [ANR FLFS & MINES grant, until Nov. 10]
Abdelhakim Hannousse [REGIONAL COUNCIL & MINES grant, since Oct. 08]
Guilhem Jaber [NORMAL SUP grant, since Sept. 10]
Mayleen Lacouture [IST AMPLE grant, since Oct. 08]
Guillaume Le Louët [MINES grant, since Oct. 10]
Ismael Mejía [MINES grant, since Jan. 09]
Syed Asad Ali Navqi [MINES & UNIV. OF LANCASTER grant, since Nov. 08]
Hien Nguyen Van [CIFRE ORANGE grant, since Jan. 08]
Angel Núñez [MINES & IST AMPLE & ATER POLYTECH grant, since Oct. 06]
Rémy Pottier [ANR SELFXL grant, since Apr. 09]
Flavien Quesnel [MINES grant, since Oct. 09]
Jurgen Van Ham [ARMINES grant, since Sept. 10]

Post-Doctoral Fellow

Marc Léger [ANR SELFXL grant, until Oct. 10]

Administrative Assistants

Hanane Maaroufi [Part-time (50%)]
Nadine Pelleray [Part-time (33%)]

2. Overall Objectives

2.1. Presentation

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

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

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

2.2. Highlights

This year two groups of results of the ASCOLA team are particularly notable. First, in the domain of Cloud-based infrastructures, the team has shown how to harness virtual machine migration for the optimization of energy consumption [28]; Fabien Hermenier has received the award "Exceptional PhD" of University of Nantes for his work underlying these results. Three national projects on energy-aware software composition on the operating system and infrastructure levels have been started or selected.

Furthermore, the team has introduced two original foundational calculi for AOP. The Aspect Join calculus [34] constitutes the first formal definition of distributed aspect languages that allows the proof of properties (in our case using bisimulation) over distributed applications that are subject to modifications through aspects. The A calculus [19] supports the to-date most comprehensive aspect model for object-oriented base languages and supports correctness properties for a wide range of constructs using a parametrized calculus and type system.

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, ADLs and services), aspects, and DSLs. 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 [75]. 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 [84], [57]. 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 [74].

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 nowadays 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 [79].

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* [87] of a software system describes the system as a composition of *components* and *connectors*, where the connectors capture the *interaction protocols* between the components [49]. 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 [76]. 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* [59] 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 [88], [71] 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 [70], [47] 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 [69], 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 [89]. 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 [56]. 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 [91]) 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* [60], [61], 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 [83], [51], [63], provide more concise formal semantics for aspects and enable analysis of their properties [50], [62], [60], [48]. 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 [82], component interactions [92], [85] and service orchestrations [58] 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 [54], [65].

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 [68], or context-free or turing-complete expressiveness [86], [53]. 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 [60] and non-regular ones [90], [81]. 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 [80], connectors between (complex) software artifacts [77], or (based on, possibly concretizations of, *design patterns* [67]) 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* [64] as well as the large set of so-called *skeletons* [55], 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 [78], [93].

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* [52], [73]. 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 [72] and the modularization of the distribution concern itself [89]. 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 [66] 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 [50], [63].

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.

In 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 Sections 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.

In 2010, we mainly provided new results about the QoS properties in service-oriented architectures [6] and the evolution of such systems in the context of the CESSA research project, see 7.2.

Our current work is based on the following observation: similar to other compositional structuring mechanisms, SOAs are subject to the problem of cross-cutting 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 cross-cutting 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 and Grid computing

A cluster is a group of coupled computers that work together closely through fast Local Area Networks. Clusters are usually deployed within one administration domain to improve performance (for scientific applications) or availability (e.g., for Internet services hosted by a data center) compared to a single computer configuration. A grid is a collaboration between different administrative domains that share a part of their infrastructure. It is composed of a set of nodes which could be of very different nature, like clusters or (low-bandwidth wide-area) networks of personal computers or super-computers. 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 extended the Entropy framework to allow the implementation of advanced scheduling policies though efficient cluster-wide context switches of virtualized jobs (i.e job embedded in one or several virtual machines) across a cluster.

We continued to deal with self-administration of large distributed architectures with the partners of the ANR SelfXL project, see Sec. 7.2. 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. 7.3.

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.

This year, as part of understanding the potential target applications of our new language ECaesarJ (see Sec. 5.2), we have shown how a proper combination of advanced features inherited from Object-Oriented, Aspect-Oriented, and Event-based Programming could support the programming of context-aware applications.

5. Software

5.1. Awed

Participants: Mario Südholt [correspondent], Ismael Mejía.

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 the AWED model has been used to implement new results on the evolution of grid algorithms by means of invasive distributed patterns (see. Sec. 6.1) and employed in the CESSA project proposal (see Sec. 7.2) as a basis for our work on the secure evolution of service-oriented architectures. Furthermore, an application of the AWED system to the adaptation of a satellite-based automatic tolling system that we have developed in the context of an industry cooperation with Siemens AG, Munich, Germany, has been presented in the journal "IEEE Computer" in 2010, see Sec. 6.1.

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

5.2. ECaesarJ

Participants: Jacques Noyé [correspondent], Angel Núñez.

ECaesarJ is a language developed in the context of the European project AMPLE (see Sec. 7.3), 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 the TUD), but also *polymorphic events* and *state machines*. Unlike AspectJ, ECaesarJ does not include a class-like concept of aspect. Instead, it deals with pointcuts and advices 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.

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.

Finally, 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.

ECaesarJ is available on request under the GPL license

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 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 pass 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 interact with Entropy and can be used to define administrative placement rules.

In 2010, Entropy has been presented at the CloudComp conference, and has 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], Marc Léger, Mahmoud Ben Hassine.

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 [25].

As part of the Galaxy project (see Sec. 7.2), we have developed with the INRIA Adam project-team an adaptation of FPath/FScript to FraSCAti, a component framework providing runtime support for the Service Component Architecture (SCA). In that way, software architects are able to navigate using FPath notation through FraSCAti architectures and to reconfigure them with FScript.

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

5.5. WildCAT

Participants: Thomas Ledoux [correspondent], Mahmoud Ben Hassine.

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.

In the context of the Galaxy project (see Sec. 7.2), we have shown how WildCAT can be used to monitor a SCA application running in OW2 FraSCAti runtime.

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 two subject matters: aspect languages and their application to distributed systems, as well as the foundations of aspects.

6.1.1. Aspect languages, distribution and applications

We have developed support for aspect languages for distributed systems, in particular to support expressive scoping strategies for dynamic evolution of distributed systems, have proposed means to integrate notions of distributed events and aspects in an object-oriented context, and have developed aspect-based support for the invasive composition of distributed programming patterns.

- **Scoping strategies for distributed aspects.** We have developed scoping mechanisms to configure and deploy distributed aspects [15]. These mechanisms enable us to reuse aspects in different contexts and to ensure properties (for example, an aspect should not migrate from one host to another). The operational semantics of our scoping features has been precisely defined with interpreters. This work has been done by members of the associate team RAPIDS (see Sec. 7.4).
- **Object-oriented events with EScala.** We have further refined the notion of events initially introduced in ECaesarJ [42] (see Sec. 5.2). These object-oriented events combine traditional imperative events, triggered explicitly, with the declarative events of aspect-oriented programming, which can be seen as declarative queries over implicit events. An efficient and type-safe implementation of the concept has been realized as an extension of Scala. An extension of this work has been accepted for publication at AOSD'11.

- **Invasive patterns for distributed programming.** We have developed a more structured approach to the composition of complex software systems in order to improve their modularization and evolution properties [26]. Concretely, we have provided two contributions, we (i) have presented a small kernel composition language for structured gray-box composition with explicit control mechanisms for invasive access to software entities and a corresponding aspect-based implementation; (ii) have presented and compared evolutions using this approach to gray-box composition in the context of two real-world software systems: benchmarking of grid algorithms with NASGrid and transactional replication with JBoss Cache.

Furthermore, we have produced several results on the use of AOSD in large-scale applications, in particular, an analysis of the use of AOSD in industrial applications.

- **AOSD in practice.** Together with partners from industry and academia we have published an overview article on the current state of AOSD in practice [13] that presents, in particular, a set of large-scale real-world applications using AOP. We have contributed through the AWED approach (see Sec. 5.1) to distributed aspects in the context of a satellite-based automatic tolling application whose modularization properties we have improved with Siemens AG, Germany.
- **Systems-of-systems.** This year we have started to consider composition issues in systems-of-systems [30]. Concretely, we have studied integration problems in such systems (see Sec. 6.2 for details) and devised a high-level aspect-based approach to address these problems.

Finally, Mario Südholt has participated in the publication of the proceedings of the international conference of AOSD [38] as PC chair and principal co-editor.

6.1.2. Foundations of aspects

We have contributed two major results to the foundations of aspect-oriented programming: a new calculus that largely integrates aspect-oriented and object-oriented concepts and the first foundational calculus for distributed aspects. Furthermore, we have presented a new notion of views on component-based systems.

- **The A Calculus.** With partners from Vrije Universiteit Brussel and Aarhus University, we have proposed a new foundational calculus for AOP that supports the most general aspect model to-date compared to existing calculi and the deepest integration with plain OO concepts [19]. Integration with OOP is achieved essentially by modeling `proceed` using first-class closures. Two well-known pointcut categories, *call* and *execution* that are commonly considered similar are shown to be significantly different; our calculus enables the resolution of the associated soundness problems. Our calculus also includes *type ranges*, an intuitive and concise alternative to explicit type variables that allows advices to be polymorphic over intercepted methods. Finally, our calculus if the first aspect calculus to use calculus parameters to cover type safety for a wide design space of other features. The soundness of the resulting type system has been verified in Coq.
- **The Aspect Join Calculus.** We have developed the first formal theory of distributed aspects [34]. Based on the distributed and objective join calculus, our calculus is presented with a (chemical) operational semantics and a type system. We have also defined a translation of the aspect join calculus into the core join calculus and shown the correctness of the translation with a proof of bisimilarity. In this way, we provide a well-defined version of a weaving algorithm that constitutes, e.g., a main step towards an implementation of the aspect join calculus directly in JoCaml.
- **Views on component-based systems.** We have introduced a notion of views in component-based system, in particular the Fractal component model, in order to support multiple architectures [22]. A main characteristics of this notion of views is that a component in one view may correspond to an aspect in another view, thus enabling intricate relationships between components and functionality to be made explicit. A corresponding weaving algorithm generates the complete system and enables checking of protocol properties with the model checker UPPAAL.

6.2. Software Composition

Participants: Pierre Cointe, Rémi Douence, Kelly Garcés, Hervé Grall, Mayleen Lacouture, Thomas Ledoux, Marc Léger, Abdelhakim Hannousse, Ismael Mejía, Syed Asad Ali Navqi, Jean-Claude Royer, Mario Südholt.

The Ascola team has provided results mainly in two different fields: advanced composition models and model transformations.

6.2.1. Advanced composition models

We have produced several results on the dynamic reconfiguration of component-based systems and service compositions.

- **Reliable Dynamic Reconfigurations.** We have proposed a definition of consistency for (re)configurations in Fractal component architectures with a model based on integrity constraints (e.g., structural invariants). Reliability of reconfigurations is ensured thanks to a transactional approach which allows us both to deal with error recovery and to manage distributed and concurrent reconfigurations in Fractal applications [25].
- **Service composition.** In the context of Service-Oriented Computing, we have proposed a pivot architecture built around a universal language for manipulating resources [24]. It aims at solving composition problems related to adaptation, integration and coordination of services.
- **Invasive distributed patterns.** We have developed a more structured approach to the composition of complex software systems using invasive patterns for distributed programming and applied it to a service-based application [26], see 6.1 for details.
- **Systems-of-systems.** With partners from Lancaster University, we have started to consider composition issues in systems-of-systems [30]. Concretely, we have studied three integration problems of such systems: managerial independence, interface incompatibility, and component system complexity. We have provided a high-level aspect-based approach to address these problems.
- **Composition of aspects and components.** Together with partners from industry and academia we have published an overview article on the current state of AOSD in practice [13], see Sec. 6.1 for details. We have also investigated AOP in a component based context [22], see Sec. 6.1 for details.

We have also tackled several composition problems of other software artifacts or dynamical systems.

- **Compositional control for constraint solvers.** We are investigating [40] how to compose a search strategy and a constraint solver. This work shows how to build a monadic program in Haskell to control constraint propagators in C++ for the state estimation of multi-model (hybrid) dynamical systems, subject to partial and uncertain measurements.
- **Fixed Point Combinator.** We have proposed a new method [29] to characterize the fixed points described in order-theoretic fixed point theorems (Tarski, Bourbaki-Witt) and used in semantics. The method is deductive: the fixed points are "proved" in some inference system defined from deduction rules. It is a first step towards a formalization of the connection between the traditional iterative method resorting to ordinals and the original impredicative method used by Tarski.
- **Synchronization of stochastic systems.** The functional role of synchronization has attracted much interest and debate: in particular, synchronization may allow distant sites in the brain to communicate and cooperate with each other, and therefore may play a role in temporal binding, in attention or in sensory-motor integration mechanisms. In [14], we study another role for synchronization: the so-called "collective enhancement of precision." We argue, in a full nonlinear dynamical context, that synchronization may help protect interconnected neurons from the influence of random perturbations (intrinsic neuronal noise) which affect all neurons in the nervous system.

6.2.2. Model transformation

We have produced several results on the application of model-driven techniques to software product lines and on the problem of model matching.

- **Fine-grained configuration of software product lines.** We have introduced the Fiesta toolkit [16], a set of tools to assist product-line architects and product designers in the creation of fine-grained configurations of products in model-driven software product lines.
- **Constraints for product line derivation.** We have shown how to use constraint programming to derive model-driven software product lines [17].
- **Model matching.** We have proposed a mechanism to adapt matching algorithms to the ontology context, for example, to the Ontology Alignment Evaluation Initiative [21].
- **Evolution of models.** We have also provided a detailed comparison of a representative sample of model migration tools - AML, COPE, Ecore2Ecore and Epsilon Flock - using common migration examples [33].
- Kelly Garcés PhD thesis [11] has provided new results on heuristics-based model matching based on underlying quality measures. Her approach extracts a large set of modeling test cases from model repositories, and uses megamodels to automate the evaluation of strategies over them. To validate the approach, she has developed an AML DSL on top of the AmmA model transformation platform. She also has contributed three use cases that show the applicability of matching to interesting MDE topics: model co-evolution, pivot metamodel evaluation, and model synchronization.

6.3. Cluster, grid and cloud computing

Participants: Frederico Alvares, Gustavo Bervian Brand, Thomas Chavrier, Pierre Cointe, Rémi Douence, Fabien Hermenier, Adrien Lèbre, Thomas Ledoux, Marc Léger, Guillaume Le Louët, Jean-Marc Menaud, Hien Nguyen Van, Jacques Noyé, Rémy Pottier, Flavien Quesnel, Mario Südholt.

Our main results concern the management and the execution of the applications leveraging virtualization capabilities on cluster, grid and cloud infrastructures.

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

6.3.1. Virtualization and Job management

With respect to virtualization and job management, the originality of our work is twofold:

- The use of several control loops to manage both applications and physical resources in an autonomic way.
- A strong synergy between applications and resources management systems to improve the decision processes of each loop.

Concretely, we have achieved the following results:

- **Context switch.** We have extended the Entropy framework and proposed the concept of "cluster-wide context switch" [23], a building block leveraging virtualization capabilities to facilitate advanced scheduling implementations. Such a context switch relies on the manipulation of virtualized-jobs (vjobs), i.e. jobs composed of VMs, through their life cycle.
- **Resource management.** In cooperation with the Myriads project-team from INRIA Rennes-Bretagne Atlantique, we have addressed resource-management issues using virtualization [20]. We have proposed a way to improve best-effort management in grids through the Saline framework and extended this work through the integration of Entropy within the Saline framework. This integration allows us to manage VMs through several sites in an efficient way [12] Finally, we have suggested

a new approach aiming at automatically adapt both hardware and software resources to the applications' needs 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)* [41].

- **VMScript.** In the context of the Entropy platform for task consolidation in clusters, we have developed VMScript, a domain specific language for the administration of virtualized grid infrastructures [32]. This language relies on set manipulation and is used to introspect physical and virtual grid architectures thanks to query expressions and to modify the placement of VMs.
- **Invasive patterns.** We have shown how invasive patterns for distributed programming can be used to more declaratively define and implement evolution problems of grids [26], see Sec. 6.1 for details.

6.3.2. Optimization of energy consumption

We have published three results on the energy consumption on the application level.

- **Resource arbitration.** We have extended the Entropy framework [28] to address the trade-off between application performance and energy consumption and to enable arbitration of resource allocations in the case of contention.
- **Energy-aware service compositions.** We have proposed an approach for energy-aware self-adaptation of SOA-based applications [36]. The energy efficiency properties of services are defined by means of Quality of Service criteria and a set of event-condition-actions is defined to enable the application to react to environmental changes and optimize its energy consumption.
- **Energy-aware software.** We have presented a position paper to the French community of Software Engineering. The objective of this work [27] was to raise discussions around the interest of reifying the energy at the software level.

7. Contracts and Grants with Industry

7.1. Contracts with Industry

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

To satisfy QoS properties in 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 benefits 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 [28].

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

7.2. National Initiatives

7.2.1. ANR CESSA: "Compositional evolution of secure services with aspects"

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 June 2009 for funding amounting to 290 KEUR for ASCOLA from December 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, and set up an SAP community blog.

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

7.2.2. ANR/Emergence Entropy

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

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

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

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

7.2.3. ANR FLFS: "Families of Languages for Families of Systems"

Participants: Pierre Cointe [coordinator], Kelly Garcés.

FLFS involved three INRIA research teams ASCOLA, AtlanMod, and Phoenix until January 2010. Its purpose was to place domain expertise at the center of software development process. FLFS addressed the current limitations of software engineering with regarding large-scale software production, robustness, reliability, maintenance, and evolution. Our main contribution was to parametrize software development process with a specific domain of expertise.

Several domain specific languages have been modeled and implemented in the fields of Internet telephony services and pervasive computing. FLFS has applied the DSL approach to model driven engineering itself. Following the works on the ATL transformation language and on model weaving, FLFS allowed the design and implementation of the AML DSL to manage software evolution. The AML presentation, its integration in the Eclipse/AMMA platform dedicated to models transformation, several use cases has been developed in Kelly Garcés PhD defended in September 2010 [11]. FLFS results have also been presented at the Grand Colloque STIC.

7.2.4. FUI Cool-IT

Participants: Jean-Marc Menaud, Adrien Lèbre.

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

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

To this end, the COOL IT project aims at developping processes for cooling computer servers, optimizing the servers' power chain supply, implementing tools and methods of collecting energy data in real time, specifying methods for controlling the data centers consumption as a tradeoff between the computational power needed, the availability, and the energy consumption.

7.2.5. ADT Galaxy

Participants: Thomas Ledoux, Mahmoud Ben Hassine.

The technology development action (ADT) Galaxy (http://galaxy.gforge.inria.fr) has been created in order to leverage INRIA's multiple software contributions to the field of SOA (Service-Oriented Architecture). The objective of the Galaxy project is to pre-assemble technological bricks from various teams and projects, and to prepare them to be transferred through the open source software channel.

Galaxy makes it possible to design, deploy, run, monitor systems, following concepts and paradigms inherited from service-oriented, business process and dynamic architectures, and to offer a set of management functions for agile and dynamic systems. Most of the Galaxy technologies are compliant with the Eclipse and the SCA standards. The INRIA technologies Fractal, FraSCAti and GCM-ProActive are the technological drivers of this ADT.

The ASCOLA project-team provides the DSL FScript and the monitoring framework WildCAT as sub-components for building the target agile platform. For the last year, in cooperation with the Adam project-team, we have proposed an extension of FScript for FraSCAti, an implementation of the SCA standard. We have also proposed a major revision of WildCAT with several use cases and documentation at http://wildcat.ow2.org.

The duration of the project was 28 months, ending October 2010. Contributors to this ADT were mainly research project-teams, including ADAM, ECOO, OASIS, ASCOLA, TUVALU, SARDES and TRISKELL.

7.2.6. ANR/ARPEGE SelfXL

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

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

7.3. European Initiatives

7.3.1. SCALUS, "SCALing by means of Ubiquitous Storage"

Participants: Adrien Lèbre, Mario Südholt.

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 the 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), EMNantes/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. ..

7.3.2. 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/

7.4. International Initiatives

7.4.1. Associate Team RAPIDS

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

RAPIDS is an INRIA Associated Team started in January with the PLEAID Laboratory at the University of Chile, with a kickoff meeting in March in Nantes. RAPIDS stands for Reasoning about Aspect-oriented Programs and securIty in Distributed Systems. 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. Some initial results of the project are reported in Sec. 6.1 and more details are available on the RAPIDS website http://rapids.gforge.inria.fr.

8. Dissemination

8.1. Animation of the community

8.1.1. Animation

ACM/SIGOPS: Jean-Marc Menaud has been the vice-chair of the French ACM/SIGOPS Chapter (ASF) since March 2008.

AOSD: M. Südholt was program chair of the international conference AOSD'10 and helped organizing it in Rennes and St. Malo.

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.

DSAL: Jacques Noyé has co-organized the 5th international workshop on "Domain-Specific Aspect Languages" (DSAL 2010) co-located with AOSD 2010 in Rennes.

EuroSys: J.-M. Menaud is a member of the EuroSys 2010 organization committee.

JTE-Virtualization: Adrien Lèbre has organized a "Journée Thèmes Emergents" on the Virtualization in Distributed Architectures, on June 17-18, at the Ecole des Mines in Nantes, http://www.emn.fr/z-info/ascola/doku.php?id=internet:jte-virtualization-2010.

LMO: Jean-Claude Royer was program chair of LMO 2010 in Pau, http://cal-idm-lmo-gpl-2010.univ-pau.fr/lmo.html.

SPL: Jean-Claude Royer was program chair of the "Journée lignes de produits logiciels", October 20 in Paris, https://sites.google.com/site/journeespl/home.

8.1.2. Steering, journal, conference committees

P. Cointe was a member of the NOTERE 2010 conference and of the 2010 Dynamic Languages Symposium (DLS).
A. Lèbre was a member of the program committee of the 5th IEEE International Conference on Networking, Architecture, and Storage (NAS 2010), the 4th Workshop on System-level Virtualization for High Performance Computing (HPCVirt 2010, co-located with the ACM SIGOPS EuroSys conference) and the 2nd Workshop on Energy Efficient Grids, Clouds and Clusters (E2GC2 2010). He has been invited to the 4th Workshop on Virtualization Technologies in Distributed Computing (VTDC 2010, colocated with the ACM HPDC conference) to deliver a keynote on virtualization concerns in the Grid'5000 infrastructure.
T. Ledoux was a member of program committees of the 1st International Workshop on Green Computing Middleware (GCM'10) and the 9th International Workshop on Adaptive and Reflective Middleware (ARM'10).
J.-M. Menaud: He is a member of the (RenPar/CFSE/Sympa) steering committees. He has served on the program committee of the IEEE/ACM International Conference on Green Computing and Communications (GreenCom2010), the 5th Workshop on Virtualization in High-Performance Cluster and Grid Computing (VHPC'10) as part of Euro-Par 2010 and 4th Workshop on System-level Virtualization for High Performance Computing (HPCVirt 2010) as part of EuroSys 2010, and the first International Workshop on Autonomic Systems enabling Green Computing (ASGC 2011) as part of The Tenth International Conference on Networks (ICN 2011).
J-C. Royer: He is a member of the editorial board of the journal TSI. He was a member of the program committee of CAL 2010 and the workshop MDPLE.
M. Südholt is a member of the steering committees of the international conferences AOSD and Software composition. He is also member of the editorial board of the international journal "Transactions of AOSD", which is published by Springer Verlag.

8.1.3. Thesis committees

P. Cointe has taken part in Benoit Baudry HDR (Rennes, 10 Dec.) and Kelly Garcès PhD (Nantes, 28 Sept.) committees.
J-C. Royer was reviewer of three PhDs: Idrissa Abdoulaye Dieng (Université de Grenoble, May 31), Fabien Bonnefoy (Université Pierre et Marie Curie, September 27), Mario Sànchez Puccini (Bruxelles VUB, December 15).
M. Südholt has taken part in three PhD evaluation committees: Oscar Gonzales (VU Brussel, Belgium and U. Los Andes, Colombia; 17 July), Fan Yan (TU Denmark, 25 Oct.), Laurent Huber (Irisa, 17 Dec.).

8.1.4. Evaluation committees and expertise

P. Cointe was a member of the AERES visiting committee for LIRMM and IRCAM. He served in four selection/hiring committees (EMN, Polytech'Nantes and University of Nantes) for associate and full professor positions.
He acted as an expert for the ANR International White Program 2010 call.
T. Ledoux was a member of selection committees for hiring new assistant professors in Computer Science at University of Rennes and University of Nantes. He was member of the selection committee for hiring new post-docs in INRIA Rennes-Bretagne Atlantique. Finally, he acted as an expert for the ANR Arpege 2010 call.
J.-M. Menaud acted as an expert for the ANR ARPEGE 2010 and ANR International White Program 2010 call.
J-C. Royer: acted as an expert for the ANR Blanc 2010 call.

8.2. Teaching

MSc ALMA: Jean-Claude Royer did a lecture to students of the ALMA master on software product lines.

M. Südholt was responsible for the AOSD specialization module (2nd year). He as well as Thomas Ledoux, J.-M. Menaud et J. Noyé taught in this module.

MSc MRI-IFSIC: Adrien Lèbre did lectures to students of the MRI master (University of Rennes).

8.3. Collective Duties

P. Cointe: He is the head of the LINA computer science laboratory (UMR 6241)http://www.lina.univ-nantes.fr/. He is a member of the board of the Doctoral School MATISSE in Rennes as well as the selection and evaluation committee of the cluster Images & Réseaux. He is member of the scientific board of the GDR GPL.

T. Ledoux: He is a member of the board of the Regional Doctoral School STIM. He is a member of the board of Follow-up committee of the PhDs thesis at LINA. Finally, he is member of the administrative committee of the INRIA Rennes-Bretagne Atlantique.

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: He has served on the CR hiring committee of INRIA Rennes, Bretagne Atlantique. Furthermore, he is a member of the council of the *Laboratoire Informatique de Nantes Atlantique* (LINA, UMR 6241). He also serves on the selection and evaluation committee of the competitiveness cluster Images & Réseaux. Finally, he is a member of the governing board of the European Network of Excellence in AOSD.

9. 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)", Slovénie Maribor, T. DE HONDT (editor), LNCS, Springer-Verlag, June 2010, 000.

[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), LNCS, 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] É. TANTER, R. TOLEDO, G. POTHIER, J. NOYÉ. *Flexible Metaprogramming and AOP in Java*, in "Science of Computer Programming - Special issue on Experimental Software and Toolkits", 2008, vol. 72, n° 1-2, p. 22–30.

Publications of the year

Doctoral Dissertations and Habilitation Theses

[11] K. GARCÉS. *Une approche pour l'adaptation et l'évaluation de stratégies génériques d'alignement de modèles*, Université de Nantes, Sep 2010, http://hal.inria.fr/tel-00532926/en.

Articles in International Peer-Reviewed Journal

[12] J. GALLARD, A. LÈBRE. *Managing Virtual Resources: Fly through the Sky*, in "ERCIM News", Oct 2010, p. 36–37, http://ercim-news.ercim.eu/en83/special/managing-virtual-resources-fly-through-the-sky, http://hal.inria.fr/inria-00536994/en.

[13] A. RASHID, T. COTTENIER, P. GREENWOOD, R. CHITCHYAN, M. REGINE, C. ROBERTA, M. SÜDHOLT, W. JOOSEN. *Aspect-Oriented Software Development in Practice: Tales from AOSD-Europe*, in "IEEE COMPUTER", Feb 2010, vol. 43, n° 2, p. 19-26 [*DOI :* 10.1109/MC.2010.30], http://hal.inria.fr/hal-00470420/en.

[14] N. TABAREAU, J.-J. SLOTINE, Q.-C. PHAM. *How synchronization protects from noise.*, in "Plos Computational Biology", 2010, vol. 6, n° 1, e1000637 [*DOI :* 10.1371/JOURNAL.PCBI.1000637], http://hal.inria.fr/inria-00460739/en.

[15] É. TANTER, J. FABRY, R. DOUENCE, J. NOYÉ, M. SÜDHOLT. *Scoping strategies for distributed aspects*, in "Science of Computer Programming", Jul 2010 [*DOI :* 10.1016/J.SCICO.2010.06.011], http://hal.inria.fr/inria-00523569/en.

International Peer-Reviewed Conference/Proceedings

[16] H. ARBOLEDA, A. ROMERO, R. CASALLAS, J.-C. ROYER. *Fiesta Toolkit: Model-Driven Software Product Lines in Practice*, in "Brazilian Conference on Software: Theory and Practice", Brésil, Sep 2010, p. 61–66, http://hal.inria.fr/hal-00536847/en.

[17] H. ARBOLEDA, V. VARGAS, F. DIAZ JUAN, J.-C. ROYER. *Automated Reasoning for Derivation of Model-Driven SPLs*, in "2nd International Workshop on Model-driven Approaches in Software Product Line

Engineering (MAPLE 2010) at SPLC 2010", Corée, République De, L. UNIVERSITY (editor), Sep 2010, vol. Volume 2, p. 181–188, http://hal.inria.fr/hal-00536845/en.

[18] N. BELDICEANU, F. HERMENIER, X. LORCA, T. PETIT. *The increasing nvalue constraint*, in "7th International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR'10)", Italie, 2010, p. 20-35, http://hal.inria.fr/hal-00485565/en.

[19] B. DE FRAINE, E. ERNST, M. SÜDHOLT. *Essential AOP: The A Calculus*, in "European Conference on Object-Oriented Programming (ECOOP'10)", Slovénie Maribor, T. DE HONDT (editor), LNCS, Springer Verlag, June 2010, 000, http://hal.inria.fr/hal-00467405/en.

[20] J. GALLARD, A. LÈBRE, C. MORIN. *Saline: Improving Best-Effort Job Management in Grids*, in "PDP 2010: The 18th Euromicro International Conference on Parallel, Distributed and Network-Based Computing – Special Session: Virtualization in Distributed Systems", Italie Pisa, 2010, http://doi.ieeecomputersociety.org/10.1109/PDP.2010.61, http://hal.inria.fr/inria-00426373/en.

[21] K. GARCÉS, W. KLING, F. JOUAULT. *Automatizing the Evaluation of Model Matching Systems*, in "Workshop on matching and meaning 2010", Royaume-Uni Leicester, 2010, http://hal.inria.fr/hal-00466946/en.

[22] A. HANNOUSSE, G. ARDOUREL, R. DOUENCE. *Views for Aspectualizing Component Models*, in "the 9th AOSD Workshop on Aspects, Components, and Patterns for Infrastructure Software (ACP4IS 2010)", France St-Malo, Mar 2010, acp4is10, http://hal.inria.fr/hal-00448686/en.

[23] F. HERMENIER, A. LÈBRE, J.-M. MENAUD. *Cluster-Wide Context Switch of Virtualized Jobs*, in "VTDC10 - The 4th International Workshop on Virtualization Technologies in Distributed Computing", États-Unis Chicago, Jun 2010, http://hal.inria.fr/inria-00476790/en.

[24] M. LACOUTURE, H. GRALL, T. LEDOUX. *CREOLE: a Universal Language for Creating, Requesting, Updating and Deleting Resources*, in "International Workshop on the Foundations of Coordination Languages and Software Architectures (FOCLASA 2010)", France PARIS, M. MOUSAVI, G. SALAÜN (editors), Sep 2010, http://hal.inria.fr/inria-00493063/en.

[25] 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, LNCS, Springer Berlin / Heidelberg, Jun 2010, p. 74-92, http://hal.inria.fr/hal-00474798/en.

[26] I. MEJIA, M. SÜDHOLT. *Structured and flexible gray-box composition: application to task rescheduling for grid benchmarking*, in "IADIS International Conference APPLIED COMPUTING 2010", Roumanie Timisoara, International Association for Development of the Information Society and "Politehnica" University of Timisoara, Romania, 2010, http://hal.inria.fr/inria-00511843/en.

[27] J.-M. MENAUD, A. LÈBRE, T. LEDOUX, J. NOYÉ, P. COINTE, R. DOUENCE, M. SÜDHOLT. *Vers une réification de l'énergie dans le domaine du logiciel*, in "Journées du GDR Génie de la Programmation et du Logiciel", France, March 2010, 000, http://hal.inria.fr/hal-00467411/en.

[28] J.-M. MENAUD, H. NGUYEN VAN, F. DANG TRAN. *Performance and Power Management for Cloud Infrastructures*, in "Cloud 2010", France Miami - USA, Jul 2010, http://hal.inria.fr/hal-00481701/en.

[29] D. MILLER, A. CARAYOL, P. RONDOGIANNIS, L. BIRKEDAL, M. CZARNECKI, H. GRALL, P. LEVY, M. MIO, K. NAKATA, A. ROMASHCHENKO, J. SCHWINGHAMMER, K. STØVRING, T. UUSTALU, P. WASZKIEWICZ. *FICS 2010*, in "7th Workshop on Fixed Points in Computer Science, FICS 2010", Tchèque, République Brno, L. SANTOCANALE (editor), Aug 2010, 89, http://hal.inria.fr/hal-00512377/en.

[30] S. A. A. NAQVI, R. CHITCHYAN, S. ZSCHALER, A. RASHID, M. SÜDHOLT. *Cross-Document Dependency Analysis for System-of-System Integration*, in "15th Monterey Workshop - Foundations of Computer Software, Future Trends and Techniques for Development (Monterey'08)", France, Springer Verlag, 2010, http://hal.inria.fr/hal-00470429/en.

[31] S. A. A. NAQVI, R. CHITCHYAN, S. ZSCHALER, A. RASHID, M. SÜDHOLT. *Cross-Document Dependency Analysis for System-of-System Integration*, in "Proceeding of the 15th Monterey Workshop - Foundations of Computer Software, Future Trends and Techniques for Development (Monterey'08)", Springer-Verlag, 2010.

[32] R. POTTIER, M. LÉGER, J.-M. MENAUD. *A Reconfiguration Language for Virtualized Grid Infrastructures*, in "10th IFIP international conference on Distributed Applications and Interoperable Systems (DAIS)", France, 2010, vol. 6115, http://hal.inria.fr/hal-00474647/en.

[33] L. ROSE, M. HERRMANNSDOERFER, J. WILLIAMS, D. KOLOVOS, K. GARCÉS, R. PAIGE, F. POLACK. *A Comparison of Model Migration Tools*, in "Proc. of Models 2010 Foundation Track", Norvège, Oct 2010, http://hal.inria.fr/hal-00499395/en.

[34] N. TABAREAU. *A theory of distributed aspects*, in "9th International Conference on Aspect-Oriented Software Development (AOSD '10)", France Rennes, Saint-Malo, ACM, 2010, p. 133–144 [*DOI :* 10.1145/1739230.1739246], http://hal.inria.fr/inria-00423996/en.

National Peer-Reviewed Conference/Proceedings

[35] N. BELDICEANU, F. HERMENIER, X. LORCA, T. PETIT. *La contrainte Increasing NValue*, in "JFPC 2010 - Sixièmes Journées Francophones de Programmation par Contraintes", France Caen, Jun 2010, p. 61-70, http://hal.inria.fr/inria-00520296/en.

Workshops without Proceedings

[36] F. G. ALVARES DE OLIVEIRA JR., T. LEDOUX. *Self-optimisation of the energy footprint in Service-Oriented Architectures*, in "1st International Workshop on Green Computing Middleware", Inde, Nov 2010, http://hal.inria.fr/hal-00534607/en.

[37] G. JABER, N. TABAREAU. *Krivine realizability for compiler correctness*, in "Workshop LOLA 2010, Syntax and Semantics of Low Level Languages", Royaume-Uni Edinburgh, Jul 2010, http://hal.inria.fr/hal-00475210/en.

Scientific Books (or Scientific Book chapters)

[38] *Proceedings of the 9th Int. Conference on Aspect-Oriented Software Development*, ACM, Mar 2010, 230, http://hal.inria.fr/hal-00467398/en.

[39] E. CARIOU, J.-C. ROYER. *Langages et Modèles à Objets*, Université de Pau et des pays de l'adour, Mar 2010, ISSN 2105-102X, http://hal.inria.fr/hal-00536035/en.

Research Reports

[40] G. CHABERT, R. DOUENCE. *Controlling Contractors with Monads for Hybrid Dynamical Systems*, INRIA, Nov 2010, n^{o} RR-7451, http://hal.inria.fr/inria-00536614/en.

[41] J. GALLARD, G. VALLÉE, T. NAUGHTON, A. LÈBRE, S. SCOTT, C. MORIN. *Architecture for the Next Generation System Management Tools for Distributed Computing Platforms*, INRIA, May 2010, n^{o} RR-7325, http://hal.inria.fr/inria-00494328/en.

[42] V. GASIŪNAS, L. SATABIN, M. MEZINI, A. NÚÑEZ, J. NOYÉ. *Declarative Events for Object-Oriented Programming*, INRIA, May 2010, n^{o} RR-7313, http://hal.inria.fr/inria-00494645/en.

Other Publications

[43] J. COHEN, R. DOUENCE. *Views, Program Transformations, and the Evolutivity Problem*, 2010, 25 pages, http://hal.inria.fr/hal-00481941/en.

[44] H. GRALL. *Proving Fixed Points*, 2010, http://hal.inria.fr/hal-00507775/en.

[45] H. GRALL, N. TABAREAU. *Linear logic as a foundation for service-oriented computing*, 2010, http://hal.inria.fr/inria-00473854/en.

[46] N. TABAREAU. *Aspect Oriented Programming: a language for 2-categories*, 2010, http://hal.inria.fr/inria-00470400/en.

References in notes

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

[48] 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.

[49] 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.

[50] 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", LNCS, 2001, vol. 2192, p. 187–209.

[51] 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.

[52] 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.

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

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

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

[56] 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.

[57] 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.

[58] 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.

[59] 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.

[60] 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)", LLNCS, Springer-Verlag, October 2002, vol. 2487, p. 173–188, ftp://ftp.inria.fr/INRIA/publication/publi-pdf/RR/RR-4435.pdf.

[61] 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.

[62] 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", LNCS, Springer-Verlag, 2001, vol. 2192, p. 170–186.

[63] 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.

[64] 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.

[65] 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.

[66] 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.

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

[68] 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.

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

[70] 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.

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

[72] 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), LNCS (Lecture Notes in Computer Science), Springer-Verlag, 2002.

[73] 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.

[74] 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.

[75] 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.

[76] 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.

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

[78] 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.

[79] 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.

[80] 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.

[81] 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.

[82] 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.

[83] 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.

[84] 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.

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

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

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

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

[89] 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.

[90] 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.

[91] 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.

[92] 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.

[93] 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.