INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team ASCOLA

ASpect and COmposition LAnguages

Rennes - Bretagne-Atlantique

Theme : Distributed Systems and Services

Activity Report

2009

Table of contents

ASCOLA is a joint project between École des Mines de Nantes (EMNantes) and INRIA.

1. Team

Research Scientist
Thomas Ledoux [CR1 INRIA on leave from EMNantes]
Nicolas Tabareau [CR2 INRIA, since Sep. 09]

Faculty Member
Pierre Cointe [Professor, EMNantes, HdR]
Rémi Douence [Associate Professor, EMNantes]
Hervé Grall [Associate Professor, EMNantes]
Jean-Marc Menaud [Associate Professor, EMNantes]
Adrien Lèbre [Associate Professor, EMNantes]
Jacques Noyé [Vice head, Associate Professor, EMNantes]
Jean-Claude Royer [Professor, EMNantes, HdR]
Mario Südholt [Head, Associate Professor, EMNantes, HdR]

PhD Student
Fabricio de Alexandria Fernandes [CAPES grant from Brazil, since Oct. 06]
Frederico Alvares [MESR grant, since Oct. 09]
Hugo F. Arboleda Jimenez [MINES & Los Andes (Bogota, Colombia) University grant, until Oct. 09]
Luis Daniel Benavides Navarro [MINES grant, until Jan. 09]
Simplice Djoko Djoko [INRIA grant shared with the project PopArt, until Jun. 09]
Abdelhakim Hannousse [REGIONAL COUNCIL & EMN grant, since Oct. 08]
Fabien Hermenier [EMN grant, until Nov. 09]
Kelly Garcés [ANR FLFS & EMN grant, since Oct. 07]
Ismael Mejía [MINES grant, since Jan. 09]
Angel Núñez [MINES & IST AMPLE grant, since Oct. 06]
Rémy Pottier [ANR SELFXL grant, since Apr. 09]
Flavien Quesnel [MINES grant, since Oct. 09]
Asad Syed [MINES and Lancaster university grant, since Nov. 09]

Post-Doctoral Fellow
Nicolas Anquetil [IST AMPLE grant, until Jul. 09]
Mahmoud Ben Hassine [INRIA (together with TRISKELL on project GALAXY), since Nov. 08]
Marc Léger [ANR SELFXL grant, since Apr. 09]

Administrative Assistant
Diana Gaudin [Part-time (33%)]
Élodie Chahbani [Part-time (50%), until Nov. 09]

2. Overall Objectives

2.1. Presentation

The ASCOLA project-team, created on 1 Jan. 2009, 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 semantics, static analysis of properties, etc., of these languages 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 by harnessing composition languages that manipulate expressive, in particular non-regular, interaction protocols. Furthermore, we foster the transformational development of implementations from higher-level architectural software representations using domain-specific languages that manipulate architectural patterns. Finally, we focus on abstractions and development methods for distributed and concurrent applications.

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.

In 2009, an action on energy efficiency in large scale distributed architectures, from the system to the application level, has been created. This action involves three permanent staff: Adrien Lèbre, Thomas Ledoux and Jean-Marc Menaud. A first result concerns the national price on Green IT ”Prix de la croissance verte numérique” (http://www.prixdelacroissancevertenumerique.eu) delivered for the work done around the Entropy framework.

2.2. Highlights of the year

Two groups of results of ASCOLA deserve particular notice:

- ASCOLA participates in the European training network SCALUS that started end of 2009 and involves 11 academic and industrial partners working on a foundation for ubiquitous storage systems that can be scaled with respect to multiple characteristics, such as capacity, performance, distance, security (for details see http://ralyx.inria.fr/2008/Raweb/obasco/uid63.html)

  Furthermore, we have set up and will coordinate the national project CESSA that involves two academic and two industrial partners on the topic of the secure evolution of service-oriented architectures using new composition mechanisms, in particular aspects (for details see Sec. 8.2).
- We have developed new techniques for the virtualization of large-scale applications running on cluster architectures. More specifically, we have proposed new techniques for the placement of activities within clusters based on constraint-based placement algorithms and developed Entropy, a corresponding framework and tool. These results have been published in the main international conference on virtualization techniques, VEE’09 [25] (for details see Sec. 6.4).

Furthermore, Nicolas Tabareau joined ASCOLA in September 2009 as a junior INRIA researcher, starting work on the foundations of programming language semantics, aspect languages (for distributed systems), and formal methods for the (certified) compilation of programming languages, especially domain-specific ones.

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, and ADLs), aspects, protocols, patterns, 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. 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 [71]. 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 [81], [52]. 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 [70].

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 [75].

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* [84] of a software system describes the system as a composition of *components* and *connectors*, where the connectors capture the *interaction protocols* between the components [44]. 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 [72]. 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* [54] 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 [85], [66] 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 [65], [42] 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 [64], 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 [86]. 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 [51]. 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 [88]) 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* [55], [56], 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 [80], [46], [58], provide more concise formal semantics for aspects and enable analysis of their properties [45], [57], [55], [43]. 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 [92], [79], component interactions [89], [82], [90] and service orchestrations [53] 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 [49], [60].

However, the limited expressive power of regular languages has led to a number of approaches using more expressive *non-regular* interaction protocols that typically provide context-free or turing-complete expressiveness [83], [48]. 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 [55] and non-regular ones [87], [77]. The modification of interaction protocols by aspects seems highly promising for the *integration of aspects and components*: since interaction protocols are part of a component interface, component implementations remain strongly encapsulated. To the contrary, since protocols make information about the implementation explicit, they allow aspects to modify certain implementation characteristics in a controlled manner. However, only few researchers have started to explore this trail [78].

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 [76], connectors between (complex) software artifacts [73], or (based on, possibly concretizations of, *design patterns* [62]) 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* [59] as well as the large set of so-called *skeletons* [50], 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.

Patterns thus enable certain types of large-scale applications to be concisely and declaratively defined at different levels of abstraction. However, their integration with other modularization mechanisms, especially aspects, has not yet been explored. Furthermore, there have been few approaches using pattern-based representations in the context of the transformational development 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 [74], [91].

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* [47], [68]. 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 [67] and the modularization of the distribution concern itself [86]. 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 [61] 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 [45], [58].

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.

We have developed new techniques to debug and test such infrastructures, see Sec. 6.4, and implemented these techniques in the context of the AWED system for explicitly distributed aspect-oriented programming, see Sec. 5.1. Furthermore, we have also improved the way in which software is modularized, localizing its variability in independent aspects as well as improving the definition of complex configuration logic to customize software product lines as part of the European project AMPLE, see Sec. 8.3.

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 2008, we mainly provided new results about the quality of services in service-oriented architectures [1]. In 2009, we focused on the application of aspects to security in service-oriented architectures (SOAs). This work has led to a proposal called CESSA that has been accepted by the french research agency ("Agence nationale de la recherche", ANR): see Sec. 8.2 for information about the project like its funding and its members.

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 [14].

We continued to deal with self-administration of large distributed architectures with the partners of the ANR SelfXL project, see Sec. 8.2. 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. 8.3.

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 (see Sec. 6.1).

5. Software

5.1. Awed

Participants: Mario Südholt [correspondent], Luis Daniel Benavides Navarro.

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 (see Sec. 6.1). It can therefore be seen as an instance of our model of Event-based AOP for distributed programming.

This year the AWED model has been integrated into the CESSA project proposal (see Sec. 8.2) 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 as part of an article that is scheduled for publication in the journal “IEEE Computer” in 2010.

AWED is available at http://www.emn.fr/x-info/awed.

5.2. ECaesarJ

Participants: Jacques Noyé [correspondent], Angel Núñez.

ECaesarJ is a new language developed in the context of the European project AMPLE (see Sec. 8.3), 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 turns 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 an asymmetric 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.

5.3. Entropy

Participants: Jean-Marc Menaud [correspondent], Fabien Hermenier, Adrien Lebre, Hien Nguyen Van, Rémy Pottier, Marc Léger.

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 and scheduler objectives.

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 [40].

Entropy has been presented at Supercomputing 2009, is tested by OrangeLabs, DGFiP (direction Générale des Finances Publiques), Bull, and 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 with respectively the navigation and the dynamic reconfigurations of Fractal architectures [17]. *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 builds upon FPath - to select the elements to reconfigure - but adds the possibility to define 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, to ensure reliability, the FScript interpreter integrates a back-end system that ensures that reconfigurations are performed using a transactional model and preserve ACID properties [15].

As part of the Galaxy project (see Sec. 8.2), we have developed 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. STSLib

Participants: Jean-Claude Royer [correspondent], Fabricio Fernandes.

STSLib is devoted to the definition, verification, and execution of a component model based on symbolic transition systems (STS). The current main functionalities of STSLib are: design of software components with STS, building of architectures, some preliminary verifications, runtime support, and visualization using graphviz and prefuse.

The main concept behind STSLib is that an STS combines the dynamic description of a system with a description of its data types. The dynamic description is based on a simple state machine and textual notation and the data part is currently provided as a Java class. STSs add to LTSs full data types, guards with receipts, input/output values and *star notations*. The star notation allows to collapse several related emissions in a concise way using a value generator. Architectures are described by textual files expressing component types, instances, communications, and exported bindings.

The library provides some basic facilities for verification based on computing the synchronous product and the configuration graph of systems. It also allows a structured view of STS, behavioral compatibility, and *event strictness* checking, see [23]. A composite is event strict iff each subcomponent is event strict and if each communication defined at the composite level occurs in the dynamic behavior of the composite. It is intended to provide tool support for designing applications based on STSs and experimenting new constructions for components and architectures.

STSLib is available at http://www.emn.fr/x-info/jroyer/WEBLIB/index.html.

5.6. 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.

This year, we have extended WildCAT to both use design and runtime events in an homogeneous way in order to monitor design evolution in the Eclipse Modeling Framework (EMF) [27].

WildCAT is an open source component used by the Galaxy project (see Sec. 8.2). It is available under GPL v2 at http://wildcat.ow2.org.

6. New Results

6.1. Aspects

Participants: Luis Daniel Benavides Navarro, Simplice Djoko Djoko, Rémi Douence, Jacques Noyé, Angel Núñez, Mario Südholt.

Our results on aspects have focused on two main activities. First we provide *expressive* support for AOP by means of configurable interpreters, model of event handling to define context, and aspect-oriented languages for distribution. Secondly, we provide *safe* support for AOP by means of operational semantics, analysis of an aspect-oriented requirement description languages, categories of aspects and restricted languages that preserve classes of properties. The two approaches are complementary (some of our work belong to both approaches) and they mostly aim at facilitating the application of AOP in real-world contexts.

6.1.1. Declarative Definition of Contexts with Polymorphic Events

A context-aware application is an application that is able to adapt its behavior in order to best meet its users' need by taking into account context information. A modular notion of context is still difficult to provide when the context depends on events happening at different places in the application. We propose a new model [29] of event handling combining explicitly triggered events with events intercepted with aspect-oriented features. The model supports event abstraction, polymorphic references to events, and declarative definition of events as expressions involving references to events from other objects. We show that this model makes it easy to define a declarative and compositional notion of event-based context. We illustrate these ideas with examples in ECaesarJ, a language with concrete support for our model (see Sec. 5.2), and relate the events of ECaesarJ to other event-handling and context-handling models.

6.1.2. Expressive Scoping of Distributed Aspects

Dynamic deployment of aspects brings greater flexibility and reuse potential, but requires proper means for scoping aspects. Scoping issues are particularly crucial in a distributed context: adequate treatment of distributed scoping is necessary to enable the propagation of aspect instances across host boundaries and to avoid inconsistencies due to unintentional spreading of data and computations in a distributed system. We extended recent work on deployment strategies for non-distributed aspects and introduced a set of high-level strategies for specifying locality of aspect propagation and activation [31]. This work shows that, given some extensions to their original execution model, deployment strategies are directly applicable to the expressive scoping of distributed aspects.

6.1.3. AWED: aspects with explicit distribution

Luis Daniel Benavides Navarro defended his PhD thesis [12] on "Distributed Aspects: better separation of crosscutting concerns in distributed software systems". He has presented the currently most comprehensive aspect model and language that enable pointcuts to be defined in terms of sequences over events occurring on different hosts, and advice to be executed on multiple hosts that are defined relative to the hosts where the corresponding pointcut has been matched or where the encompassing aspect is deployed. Furthermore, he has introduced a notion of causal distributed sequence pointcuts and shown that these pointcuts permit distributed applications to be debugged that run two orders of magnitude faster than is possible if non-distributed debugging sessions are coordinated manually — as is the case, for example, if Eclipse is used for distributed debugging. Finally, he has validated the use of distributed aspects in the context of substantial real-world middlewares, especially JBoss Cache and Apache MQ.

As part of this PhD thesis the AWED system (see 5.1) has been designed and implemented. It has been applied 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. This application has been presented as part of an article that is scheduled for publication in the journal "IEEE Computer" in 2010.

6.1.4. CALI: a common aspect language interpreter

One of the main elements of an Aspect-Oriented Programming language or framework is its pointcut language. Experimenting with AspectJ shows that two basic primitive pointcuts, call and execution, dealing with method invocation from the caller and callee standpoints, respectively, lead to confusion. This is due to a subtle interplay between the use of static and dynamic types to select execution points, dynamic lookup, and the expectation to easily select the caller and callee execution points related to the same invocation. As a result, alternative semantics have been proposed but have remained paper design. These various semantics can be reconsidered in a practical way by implementing them using CALI, our Common Aspect Language Interpreter. This framework reuses both Java as a base language and AspectJ as a way to select the program execution points of interest. An additional interpretation layer can then be used to prototype interesting AOP variants in a full-blown environment. This work [34] illustrates the benefits of applying such a setting to the case of the call and execution pointcuts. We show that alternative semantics can be implemented very easily and exercised in the context of AspectJ without resorting to complex compiler technology.

6.1.5. Property preservation in the presence of aspects

Simplice Djoko Djoko defended his PhD thesis [13] on "Aspect-oriented programming and preservation of properties" in June 2009. Aspect-Oriented Programming can arbitrarily distort the semantics of programs. In particular, weaving can invalidate crucial safety and liveness properties of the base program. In his thesis, Simplice Djoko Djoko has identified categories of aspects that preserve a larger set of classes of properties than previous approaches. It is then sufficient to check that an aspect belongs to a specific category to know which properties will remain satisfied by woven programs. These categories are defined precisely based on a language independent abstract semantics framework. The classes of properties are defined as subsets of LTL for deterministic programs and CTL* for non-deterministic ones. As part of the PhD, it has been proven that, for any program, the weaving of any aspect in a category preserves any property in the related class. He has also designed a specialized aspect language for each aspect category, which ensures that any aspect written in that language belongs to the corresponding category, and has proved that these languages preserve the corresponding classes of properties by construction.

6.1.6. Cross-document dependency analysis for system-of-system integration

Systems-of-systems are formed through integration of individual complex systems, often not designed to work together, a paradigmatic example being all systems that have to cooperate in case of a real-world emergency. A number of factors can make this integration very challenging which often leads to catastrophic failures. We have focused on three major classes of system-of-system integration problems: managerial independence, interface incompatibility, and component-system complexity. In this context, we have presented an aspect-oriented requirements description language (RDL) which uses natural language analysis capabilities to reason about dependencies across the documentation of the constituent systems of a system-of-systems [28]. The aspect-oriented compositions in the RDL also facilitate specification of cross-document constraints and inconsistency resolution strategies, which can be used for deriving proof obligations and test cases for verification and validation of the emergent behavior of a system-of-systems. We have showcased the capabilities of our RDL through a case study of a real-world emergency response system. Our analysis has shown that the querying and composition capabilities of the RDL provide valuable support for reasoning across documentation of multiple systems and specifying suitable integration constraints.

6.2. Software composition

Participants: Nicolas Anquetil, Hugo Arboleda, Rémi Douence, Fabricio Fernandes, Jean-Claude Royer.

Our results on software composition have focused on four main activities: management of multiple representation for programs, fine-grained configuration and traceability for software product lines, and providing support for extraction of components from legacy code. Some of these results strongly link ASCOLA's work on CBSE with that on AOP by expressing aspectual representations and aspectual MDE transformations. One of the main domain of investigations was software product line engineering where we have shown the benefits of component and aspect abstractions, in particular, to enable a precise management of the flow of information to properly reuse artifacts.

6.2.1. Compositional transformation of data structure representations

Achieving separation of concerns has been a core objective of software engineering for decades. In general, software can be decomposed properly only according to a single concern, other concerns crosscut the prevailing one. This problem is well known as the tyranny of the dominant decomposition. Similarly, at the programming level, the choice of a representation for a data structure frequently drives the implementation of algorithms. We have explored an alternative that forgoes the need for a dominant representation [22]. Instead, each algorithm is developed in its "natural" representation and a representation is converted into another one only if necessary. To support this approach, we designed a framework for Java, that performs partial conversions and dynamic optimizations using lazy evaluation, and that preserves soundness of executions. Through performance evaluations over graph theoretic examples we have demonstrated that this approach provides a practicable but more efficient alternative to naive, manual representation modifications.

6.2.2. Fine-grained Configurations for Software Product Lines

As part of his PhD thesis [11], Hugo Arboleda has presented an approach based on Model-Driven Development to create Software Product Lines (SPLs). In Model-Driven SPL approaches, the derivation of a product starts from a domain application model. This model is transformed through several stages reusing model transformation rules until a product is obtained. Transformations rules are selected according to variants included in configurations created by product designers. Configurations include variants from variation points, which are relevant characteristics representing the variability of a product line. As we have shown, there are at least two drawbacks in many of these approaches [20]. First, the selection of variants affects the whole domain application model, impeding fine-grained configurations, i.e. configurations at the level of individual elements in the model. Second, model transformations are coupled with variants which make their maintenance and adaptation difficult. Like in the Triskell team, our aim is to conciliate both flexibility and automation for product derivation. We define an approach that uses metamodeling and feature modeling. Our approach (1) provides mechanisms to improve the expression of variability of Model-Driven SPLs by allowing designers to create fine-grained configurations of products, and (2) integrates a product derivation process which uses decision models and Aspect-Oriented Programming facilitating the reuse, adaptation and composition of model transformation rules. We have defined a decision model as a set of aspects [21]. An aspect maintains information of what and when transformations rules that generate commonalities of products must be intercepted (joinpoints) and what transformation rules (advices) that generate variable structures must be executed instead. Our strategy maintains uncoupled variants from model transformation rules. This solves problems related to modularization, coupling, flexibility and maintainability of transformations rules because they are completely separated from variants; thus, they can evolve independently.

6.2.3. Traceability for Software Product Lines

Another problem in SPL engineering that we have addressed is the management of traceability. As part of the European AMPLE project we have created a common traceability framework across the various activities of the SPL development. We have identified four orthogonal traceability dimensions in SPL development, one of which is an extension of what is often considered as "traceability of variability" [16]. Furthermore, we have specified a metamodel for a repository of traceability links in the context of SPLs and the implementation of a corresponding traceability framework. This framework permits to perform fundamental traceability management operations, such as trace import and export, modification, query and visualization.

6.2.4. Extracting Components from Java Source Code

Architectural erosion is the process by which a system's architecture gradually degrades as maintainers make changes to the system that violates the original architectural intents. To address this problem, new languages and development methods are currently under investigation that make explicit some architectural decisions in the source code (for the benefit of the programmers) and allow automatic verification and enforcement of these decisions, either at compile or execution time. We have explored the possibilities of automatic reverse-engineering of such legacy applications in order to define a more formal model by extracting the component types they contain and to make explicit the communication channels between them [19]. This work provides an overview of the main rules and the associated tool support we have developed. Furthermore, we have defined several heuristics for the corresponding reverse-engineering task which are mainly based on communication integrity properties. If a type of interest is passed as parameter of a method or returned by a method it is classified as a data type, otherwise as a component type. The data type classification is then propagated through subtyping and also by composition. This tool is intended to help its user compare (and map) a concrete implementation with an abstract model.

6.3. Programming languages, DSLs and software development

Participants: Pierre Cointe, Kelly Garces, Hervé Grall, Mayleen Lacouture, Thomas Ledoux, Marc Léger, Jacques Noyé.

Our work have focused on four main activities. First, we have provided a DSL, named FScript, for the dynamic reconfiguration of Fractal architectures. FScript ensures the reliability of the reconfigurations thanks to a transactional approach. FScript is used in the project Galaxy. Secondly, we have participated with the AtlanMod team from EMN and INRIA in the definition of a DSL for the definition of matching strategies which compute mappings between models. The results of this work are hosted by the Eclipse foundation (see http://wiki.eclipse.org/AML). Then, we have analyzed the evolution of programming languages in the field of computer music. Finally, we have illustrated the use of coinductive definitions and proofs in big-step operational semantics using a call-by-value functional language.

6.3.1. Language support for navigation and reliable reconfiguration of Fractal architectures

Component-based systems must support dynamic reconfigurations to adapt to their execution context, but not at the cost of reliability. Open component-based systems such as Fractal provide intrinsic support for dynamic reconfiguration, but their definition in terms of low-level APIs makes it complex to write reconfigurations and ensure the reliability of the resulting systems. We have proposed FScript, a domain-specific language, to solve these issues through language support for architecture navigation and reconfiguration that facilitates the definition of reconfigurations and guaranties several properties by construction (e.g., termination of scripts by excluding the possibility of infinite loops) [17].

The FScript interpreter integrates a back-end system provided by the contributions of Marc Leger's PhD work [15]. He proposed a definition of consistency for configurations and reconfigurations in the Fractal component model. Reliability of reconfigurations has been ensured thanks to a transactional approach allowing that deals with error recovery as well as the management of concurrent reconfigurations.

6.3.2. Support for Model Evolution

We made the assumption that DSLs can serve as a bridge between application modeling and programming. As initiated as part of the FLFS ANR project (see Sec. 8.2), and capitalizing on our previous experiments in implementing DSL, we work on a methodology for developing such a DSL and for applying it to the field of model evolution. This work is done jointly with the AtlanMod team.

A first result described in [35] was the definition of the AML DSL for expressing model matching algorithms and a corresponding compiler. This DSL was used to the detection of metamodel changes. We have evaluated this implementation on two strategies including robust matching transformations from the literature.

In a second step, we have developed a three-step solution that automatically adapts terminal models to their evolving metamodels. We have evaluated this solution in terms of the accuracy and performance in the context of two externally defined metamodels well-known from the literature: the Netbeans Java metamodel and the Petrinet metamodel [24].

This work will be the kernel of K. Garces'PhD thesis as well as her work on contributing AML to Eclipse (see http://wiki.eclipse.org/AML).

6.3.3. Reflective languages, aspects and DSLs

We have analyzed the evolution of programming languages by focusing on two dialectics that we consider critical for the field of computer music [37]. On the one hand, we have studied the dialectic between ”form and opening” leading to the definition of reflective languages to deal with open-ended software architectures and aspect oriented software development to express separation of concerns. On the other hand, we have observed the tension between ”generality and speciality” as manifested by the reappearance of domain specific languages.

6.3.4. Coinductive big-step operational semantics

Using a call-by-value functional language as an example, we have illustrated the use of coinductive definitions and proofs in big-step operational semantics, enabling it to describe diverging evaluations in addition to terminating evaluations [18]. Extending their previous results [63], [69], Hervé Grall and Xavier Leroy formalize the connections between the coinductive big-step semantics and the standard small-step semantics, proving that both semantics are equivalent, when they compute not only values but also traces. They then study the use of coinductive big-step semantics in proofs of type soundness and proofs of semantic preservation for compilers.

A methodological originality of the paper is that all results have been proved using the Coq proof assistant. The authors explain the proof-theoretic presentation of coinductive definitions and proofs offered by Coq, and show that it facilitates the discovery and the presentation of the results.

6.4. Cloud, cluster and grid computing

Participants: Hien N'Guyen Van, Fabien Hermenier, Adrien Lèbre, Jean-Marc Menaud.

Large scale distributed system like grids or clusters have become increasingly popular in both academic and industrial contexts. The new cloud computing architecture approach, where computing resources are provisioned on a per-demand basis, notably to handle peak loads, instead of being statically allocated, should reinforce this trend. In this section, we present problems that we have addressed on the management of computing resources (Entropy) and data resources (kDFS) in large scale and high performant distributed systems.

6.4.1. Virtualization technologies

Virtualization technologies have recently gained a lot of interest in grid computing as they allow flexible resource management. However, the most common way to exploit grids relies on dedicated services like resource management systems (RMSs) with a static allocation of resources for a bounded amount of time. Those approaches are known to be insufficient for a high utilization of clusters or grids. To provide a finer RMS, job preemption, migration and dynamic allocation of resources are required. However, due to the development complexity of such mechanisms, advanced scheduling strategies have been rarely used in available systems.

This year, we have continued to analyze and experiment how latest VM capabilities can improve job management.

The main activities have been conducted around the Entropy framework [25]. By encapsulating each component of a job into its own VM (*vjobs*), we have extended the former proposal to combine live migration and suspend/resume capabilities: the live migration aims at adapting the assignments of VMs according to their current requirements while the suspend/resume operations provide preemption capability. Thus one can implement fine-grained scheduling policies by applying a cluster-wide context switch through the manipulation of the VMs. Thanks to this new software abstraction, developers can implement sophisticated algorithms to schedule jobs without handling the issues related to the manipulation of the VMs. They can only focus on the implementation of their algorithm to select the jobs to run while the cluster-wide context switch system performs the necessary actions to switch between VM configurations. The Entropy system has been partially redesigned to handle the cluster-wide context switch in a generic way [36], [14], [40].

Moreover, Entropy has been extended to decouple the provisioning of resources from the dynamic placement of virtual machines [32]. This resources manager aims to optimize a global utility function which integrates both the degree of SLA fulfilment and the operating costs. Results obtained through simulations validate our approach [33].

In cooperation with the Paris project-team from INRIA Rennes-Bretagne Atlantique, we have addressed two resource-management issues using virtualization.

First, we have addressed the best-effort issue in grids. To improve resource usage, most of resource management systems for grids provide a best-effort mode where lowest priority jobs can be executed when resources are idle. This particular mode does not provide any service guarantees and jobs may be killed at any time by the RMS when the nodes they use are subject to higher priority reservations. This behavior potentially leads to a huge waste of computation time or at least requires users to deal with checkpoints of their jobs. To tackle this issue, we suggested the *Saline* proposal, a generic and non-intrusive framework to manage best-effort jobs at the grid level through virtual machines (VMs) usage [41]. In Saline, each best-effort job is transparently submitted into VMs so that the computation can be relocated in another location in the grid each time the resources have been taken away by higher priority jobs. Such an approach results in better performance concerning the total execution time of best-effort requests and a large benefit according to software developments (it relieves users of the burden of implementing a specific checkpointing framework for each best-effort program).

Second, we have worked on the clarification of the different ”virtualization” solutions that are available nowadays (each providing particular functionalities). Goldberg proposed to classify virtualization techniques in two models (Type-I and Type-II), which do not allow for the classification of recent virtualization technologies. We have proposed an extension of the Goldberg model to take into account recent ”virtualization” mechanisms [26]. This proposal enables the formalization of the following terms: *virtualization*, *emulation*, *abstraction*, *partitioning*, and *identity*. We show that a single virtualization solution is generally composed of several layers of virtualization capabilities, depending on the granularity of the analysis. In this manner, the suggested model allows us to classify virtualization technologies according to their performance, similarity and portability.

6.4.2. kDFS: toward an integrated cluster file system

kDFS aims at providing an integrated cluster file system for High Performance Computing. kDFS is a distributed file system pluggable under the VFS and only based on the KDDM component of Kerrighed Single System Image (SSI). The KDDM features are used to build a cooperative cache for both data and meta-data using all available memory in the cluster. The innovating approach concerns the design and the implementation of this symmetric file system with regard to the other mechanisms available: most of the cluster management systems are designed independently without considering the benefits of strong cooperations between each service. In this project, in addition to provide the common functionalities of a distributed file system, we analyze how kDFS could exploit, cooperate with and complete the cluster services itself to improve usage and global performance.

In the context of Pierre Riteau’s Master internship achieved in 2009, we have focused on reliable execution of applications that use file systems for data storage in a distributed environment. An efficient and portable file versioning framework was designed and implemented in the distributed file system kDFS. This framework can be used to snapshot file data when a process’ volatile state is checkpointed and thereby makes it possible to restart a process using files in a coherent way. A replication model synchronized with the checkpoint mechanisms has also been proposed. It provides stable storage in a distributed architecture. The synchronization has enabled us to reduce network and disk I/O compared to a synchronous replication mechanism like RAID1 [30].

We are currently implementing a data striping policy relying on application access patterns and thus avoiding RAID alignment issues. These activities are also done conjointly with the Paris project-team. Further details are available at http://www.kerrighed.org/wiki/index.php/KernelDevelKdFS.

7. Contracts and Grants with Industry

7.1. France Telecom R&D PhD about Reliability in Fractal Architectures

Participants: Thomas Ledoux, Marc Léger.

Reliability is a main problem in systems subject to dynamic reconfigurations. The aim of Marc Leger's PhD work [15], defended in May 2009, has been to guarantee reliability of dynamic reconfigurations used to make systems evolve at runtime while preserving their availability. We have proposed a definition of consistency for configurations and reconfigurations in the Fractal component model with a model based on integrity constraints like for example structural invariants. Reliability of reconfigurations has been ensured thanks to a transactional approach allowing both to deal with error recovery and to manage reconfiguration concurrency in systems. Finally, we have proposed a modular component-based architecture so as to implement transactional mechanisms adapted to dynamic reconfigurations in Fractal applications.

The results have been used in the ANR (national project) Selfware as a self-healing foundation of autonomic distributed applications and are under development as part of the national project Galaxy, see Sec. 8.2.

This PhD work has been supported by France Telecom R&D (MAPS/AMS) for an amount of 21 KEUR.

7.2. France Telecom R&D PhD about Virtualisation in Data Center

Participants: Jean-Marc Menaud, Hien Nguyen Van.

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.

This work is supported by France Telecom R&D (Magneto) for an amount of 27 KEUR.

8. Other Grants and Activities

8.1. Regional Actions

8.1.1. MILES project / Software Engineering Cluster

This three years project funded by the *Pays de la Loire Council* ended in November 2009 and was centered on the development of new techniques in software composition, in particular, aspect-oriented programming, domain specific languages, and model engineering as well as its application to the field of real-time systems. This is joint work between teams from the LINA and IRCCyN institutes and its teams AtlanMod, COLOSS, MODAL, and STR. See also the FLFS ANR related project in Sec. 8.2.

Mario Südholt has co-coordinated the cluster and ASCOLA's funding amounts to 53 KEUR.

8.2. National Projects

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

Participants: Mario Südholt (coordinator), Hervé Grall.

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. Four partners collaborate within the project that is coordinated by ASCOLA:

- INRIA project-team ASCOLA, located at Ecole des Mines, Nantes, France,
- a security research team from Eurecom, Sophia-Antipolis, France,
- the Security and Trust team from SAP Labs, located at Sophia-Antipolis,
- IS2T, an innovative start-up company developing middleware technologies located at Nantes.

The project deals with security in service-oriented architectures. See Sect. 4.2 for details about the subject. The academic partners are highly complementary with respect to the scientific methods and techniques that are to be investigated as part of the CESSA proposal : aspect-orientation and formalization for ASCOLA, security for Eurecom. The industrial partners are also highly complementary as to their business domains, size and types of customers. As a consequence, they contribute with two well-distinct real-world use cases to the proposal: SAP propose large-scale web-based enterprise information systems while IS2T will extend its specially-tailored JVM-based execution environment.

8.2.2. ANR FLFS: "Languages family for systems family"

Participants: Pierre Cointe, Kelly Garcés.

Traditionally, software development does not rely on an in-depth knowledge of the target domain. Instead, domain-specific knowledge is integrated in the software development process in an ad hoc and partial fashion, without much formal basis or tools. In doing so, software systems are tackled in isolation, making conceptual or implementation factorization difficult. Yet, it is fundamental to observe that programs always belong to a family. In this family, they share commonalities and expose specific variations.

From a software development viewpoint, a program family represents a domain of expertise, that is, a vocabulary, notations, rules and protocols that are specific to a domain. For example, the telephony domain consists of a set of concepts, rules, protocols and interfaces that represent a precise framework to be used for the development of telephony services.

Our goal is to place domain expertise at the centre of the software development process. It is aimed to lift the current limitations of software engineering regarding large scale software production, robustness, reliability, maintenance and evolution of software components. Our key innovation is to introduce a software development process parametrized with respect to a specific domain of expertise. This process covers all the stages of software development and combines the following three emerging approaches:

- Domain-specific modelling, also known as model engineering;
- Domain-specific languages, in contrast with general-purpose languages;
- Generative programming and in particular aspect-oriented programming as a means to transform models and programs.

Our partners are the AtlanMod (J. Bézivin) and Phoenix (C. Consel) INRIA teams. The duration of the project was 36 months, ending December 2009. FLFS' main results, including the AML DSL (http://wiki.eclipse.org/AML) to express matching strategies which compute mappings between models, will be presented at the ANR STIC 2010 conference (http://colloque-stic.org/).

ASCOLA's funding part amounts to 70 KEUR. The Web page is :http://flfs.emn.fr.

8.2.3. ANR SelfXL: "Self-management of complex and large scale system"

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 infrastructures...) 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.

8.2.4. 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.

From an engineering point of view, the ASCOLA project-team provides the DSL FScript and the monitoring service WildCAT as sub-components for building the target agile platform. From a research point of view, we will participate to the definition of a new language to dynamically manipulate adaptive distributed SOA-based systems.

This year, we have presented a first prototype at the JavaOne 2009 conference and proposed an extension of WildCAT for the Eclipse Modeling Framework (EMF)[27].

Contributors to this ADT are mainly research project-teams, including ADAM, ECOO, OASIS, ASCOLA, TUVALU, SARDES and TRISKELL. The ADT lasts 28 months: the kickoff was held on July 3rd, 2008 and the project is planned to end in October 2010. The galaxy ADT is led and managed by the TUVALU project-team.

8.3. European Projects

8.3.1. MCITN 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.

8.3.2. STREP AMPLE: "Aspect-Oriented, Model-Driven Product Line Engineering"

Participants: Jean-Claude Royer, Nicolas Anquetil, Hugo Arboleda, Jacques Noyé, Angel Núñez, Mario Südholt.

The AMPLE project started in October 2006 and finished in November 2009. It involved the following partners: Lancaster University (United-Kingdom), *Universidade Nova de Lisboa* (Portugal), ARMINES/EMNantes (France), *Technische Universitat Darmstadt* (Germany), *Universiteit Twente* (The Netherlands), *Universidad de Málaga* (Spain), HOLOS (Portugal), SAP AG (Germany), Siemens AG (Germany). As part of this project, the partners have developed a Software Product Line (SPL) development methodology offering improved modularization of variations, their holistic treatment across the software life cycle and maintenance of their traceability (forward and backward) during SPL evolution. It is based on the premises that:

- Aspect-Oriented Software Development (AOSD) can improve the way in which software is modularised, localising its variability in independent aspects as well as improving the definition of complex configuration logic to customise SPLs.
- Model-Driven Development (MDD) can help expressing concerns as a set of models without technical details and support traceability of the high-level requirements and variations through model transformations.

During this third and final year of the project, we have finalized a first version of ECaesarJ, a language supporting the alignment of software product line features and their implementation (see Sec. 5.2) and considered its use in the context of building automation and customer relationship management with case studies provided by the industrial partners of the project.

We also investigated principles and tool support to manage traceability for software product lines. Four orthogonal dimensions and a general traceability metamodel were proposed. We implemented some tools to enrich the basic AMPLE traceability support, they are available at http://www.emn.fr/x-info/jroyer/AMPLE/index.html. More details can be found on the project's web page: http://www.ample-project.net.

The overall funding of the project by the European Union was 3.78 MEUR. ASCOLA's share amounted to 370 KEUR.

8.3.3. COST IC0804

Participant: Jean-Marc Menaud.

The COST IC0840 Action (Energy efficiency in large scale distributed systems) 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. Web site: http://www.cost804.org/

8.4. Collaboration with Foreign Research Groups

8.4.1. CONICYT Chile - INRIA CORDIAL Project

Participants: Jacques Noyé, Rémi Douence, Mario Südholt, Luis Daniel Benavides Navarro, Angel Núñez.

CORDIAL stands for COncuRrency, Distribution, and Interactions in Aspect Languages. The objective of this CONICYT/INRIA project, started in January 2008, was to advance the state of the art in concurrent and distributed aspect-oriented programming by leveraging the expertise of the two participating teams: ASCOLA in Nantes and the newly-created PLEIAD laboratory, led by Éric Tanter, at *Universidad de Chile*. The project was renewed in 2009 with funding limited to two visits. Thanks to extra funding Éric Tanter and Johan Fabry visited ASCOLA in January and June, respectively, while Angel Núñez and Jacques Noyé visited PLEAID in May. Some initial results of the project are reported in Sec. 6.1.

9. Dissemination

9.1. Animation of the Community

9.1.1. Animation

AOSD Summer School 2009:

The fourth Summer School on Aspect-Oriented Software Development (AOSD), organized by Rémi Douence and Thomas Ledoux from the ASCOLA team in August 2009 (see http://www.emn.fr/x-info/aosdsc09/), provided an intensive week of lectures on advanced aspect-oriented topics in the context of programming languages, analysis and design, formal methods and application domains. This summer school brought together PhD students and lecturers as well as other researchers and practitioners who were interested in aspect-oriented software development. We have welcomed 38 attendees from 4 different continents.

DSAL: Jacques Noyé has co-organized the 4th international workshop on "Domain-Specific Aspect Languages" (DSAL 2009) [38] hosted by the 8th International Conference on Aspect-Oriented Software Development (Charlottesville, VA, USA, March 2009). The DSAL workshop series aims to bring the research communities of domain-specific language engineering and domain-specific aspect design together. This fourth edition focuses on language embedding that raises specific issues for language designers, such as proper symbiosis between, and composition of, domain-specific extensions (DSXs).

SPL Day: Jean-Claude Royer organized a meeting in Nantes October 15 on "Software Factories and Software product Lines", (see http://www.emn.fr/x-di/productlines/). Thirty participants attended this event, mostly from industry.

Les jeudis de l'objet: This bimonthly industrial seminar organized by our group is now ten years old. Surviving the annual conferences Objet/OCM, it has become a great place for local industry to acquire knowledge about emerging technology, exchange ideas about state-of-the-art technologies, share experiences around the technologies associated with objects and components. Each seminar presents either a state of the art of an emerging technology (Spring, Ruby on Rails, Google Web Toolkit, etc.) or feedback on an industrial project in the field of large software architectures (mobility-based applications in a small enterprise, open source middleware...). For more details on the past/future agenda, go to http://www.emn.fr/jeudis-objet.

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

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.

9.1.2. Steering, journal, conference committees

P. Cointe: He is a member of the ECOOP and LMO steering committees (http://www.ecoop.org) as well as a member of the AOSD Summer School board. He served in the SASO and NOTERE 2009 committees

P. Cointe was a member of the AERES visiting committee for the LIRMM (december 2009). He is a member of the scientific committee of the GPL GDR (http://gdr-gpl.cnrs.fr) and a member of the validation of the Images et Réseaux (Media and Networks) cluster (http://www.images-et-reseaux.com/).

A. Lèbre: He was a program committee member of : CFSE-7 (7th French Conference on Operating Systems, Toulouse, France), E2GC2 (Energy Efficient Grids, Clouds and Clusters workshop, Banff, Canada), HPCVirt 2009 (System-Level Virtualization for High Performance Computing workshop, Nuremberg, Germany). He was one of the publicity co-chairs for the IEEE 9th International Conference on Computer and Information Technology (Xiamen, China).

T. Ledoux: was a program committee member of the international journal Annals of Telecoms (special issue "The Fractal Initiative"), the 8th Workshop on Adaptive and Reflective Middleware (ARM 2009, Urbana Champaign, Illinois, USA, December 2009).

J.-M. Menaud: He is a member of the (RenPar/CFSE/Sympa) steering committees. He has served on the program committee of CFSE-7 (7th French Conference on Operating Systems, September 2009, Toulouse, France), the 4th Workshop on Virtualization in High-Performance Cluster and Grid Computing (VHPC'09) as part of Euro-Par 2009 and 3rd Workshop on System-level Virtualization for High Performance Computing (HPCVirt 2009) as part of EuroSys 2009.

J. Noyé: He was a program committee member of LMO 2009 (*Langages et Modèles à Objets*), Nancy, March 2009, and SC 2009 (Software Composition), Zurich, July 2009. He was the co-editor of a special issue on Domain-Specific Aspect Languages published by the IET Software journal (June 2009). He is a program committee member of the special track on Object-Oriented Programming Languages and Systems at the ACM Symposium on Applied Computing (Sierre and Lausanne, Switzerland, March 2010).

J.-C. Royer: He is a member of the program committee of TSI. He was a member of the program committee of CAL 2009 and is a member of the program committee of CAL 2010. He is program chair of LMO 2010.

M. Südholt: He is a member of the steering committee of the Aspect-Oriented Software Association (AOSA), the primary sponsor of the annual *International conference on Aspect-Oriented Software Development (AOSD)*. He is also a member of the steering committee of the annual *International Conference on Software Composition*, which is co-located with the conference TOOLS Europe. Since 2009, he is member of the editorial board of the international journal “Transactions of Aspect-Oriented Software Development” that is printed by Springer-Verlag.

In 2009, he has been appointed as PC chair of the conference AOSD'10, has served as the PC chair of the international workshop FOAL.09 [39] and has acted as a workshop co-chair of ECOOP'09. Finally, he has served on the program committee of the international conference SC'09.

9.1.3. Thesis committees

T. Ledoux: was a member of the PhD committee of Marc Léger (Ecole des Mines de Paris, 19 May 09).

J.-M. Menaud: was a member of the PhD committee of Fabien Hermenier (Ecole des Mines de Nantes, 26 Nov. 09).

J.-C. Royer: was a member of the PhD committee of Hugo Arboleda (Université de Nantes and Universidad de Los Andes, 28 Oct. 09).

M. Südholt: was a member of the PhD committees of Bert Lagaisse (KU Leuven, Belgium, 15 Oct. 09), Romain Delamarre (Irisa, 2 Dec. 09) and Daniel Benavides (Ecole des Mines de Nantes, 19 Jan. 09).

9.1.4. Evaluation committees and expertise

P. Cointe: is a member of the MSTP (*Mission Scientifique Technique Pédagogique*) since March 2003. He is a member of the France-Maroc scientific committee in charge of the *Software Engineering* cooperation. He is heading the software theme of the *Media and Networks* Cluster (see http://www.images-et-reseaux.com) and the INRIA representative at the *NESSI SRA*.

J.-M. Menaud: was an expert for the ANR *Blanc International 2009* call.

J.-C. Royer: acted as an expert for the ANR DEFIS 2009 call.

M. Südholt: has acted as an expert for the Swiss and Dutch science foundations.

9.2. Teaching

MSc EMOOSE. Between September 1998 and August 2009, the team managed, in cooperation with the *Vrije Universiteit Brussel* (VUB) and the collaboration of a network of partners, an international Master of Science program EMOOSE (European Master of Science on Object-Oriented and Software Engineering Technologies). This program was dedicated to object-oriented software engineering in a broad sense, including component-based and aspect-oriented software development. The eleventh promotion graduated in August 2009. See also: http://www.emn.fr/emoose.

MSc Alma. The faculty members of the team participate in this master program. Mario Südholt is responsible for its module on Aspect-Oriented Software Development and ASCOLA members give lectures about new trends in the field of component and aspect-oriented software engineering.

MSc MRI-IFSIC. Adrien Lèbre participated in this master program done at the CS university in Rennes. He gave lectures about distributed file systems and parallel I/O issues in cluster and grids and described the interest of latest VM capabilities in cloud computing platforms.

AOSD-Europe Summer School. Rémi Douence and Mario Südholt participated in this one week school in Nantes, France (see Section 9.1.1 for details). They gave a lecture on *Formal Methods for AOP* and *Aspect-Oriented Technologies for Distributed Software*, respectively.

Introductory Course to Scala. In May, Jacques Noyé gave an introductory course to the programming language Scala at the Department of Computing of the University of Chile in Santiago.

9.3. Collective Duties

P. Cointe: He is chairman of the *Laboratoire Informatique de Nantes Atlantique* (LINA, UMR 6241) grouping together the University of Nantes, the Ecole des Mines de Nantes and the CNRS. Pierre is the academic chairman of the *Software Engineering theme* of the Images et Réseaux cluster. Pierre was member of the hiring committee for the EMNantes-INRIA chaire.

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.

J.-M. Menaud: He is deputy of the EMNantes computer science departement in charge of the training program.

M. Südholt: In 2009, 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 is also member of the governing board of the Int. Association of AOSD.

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", vol. 5364, Springer-Verlag, December 2008, 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, vol. 5346, Springer Verlag, December 2008, p. 183-202.

[3] L. D. BENAVIDES NAVARRO, M. SÜDHOLT, W. VANDERPERREN, B. DE FRAINE, D. SUVÉE. *Explicitly distributed AOP using AWED*, in "Proc. of the 5th ACM Int. Conf. on Aspect-Oriented Software Development (AOSD'06)", ACM Press, March 2006, p. 51-62.

[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] R. DOUENCE, T. FRITZ, N. LORIANT, J.-M. MENAUD, M. SÉGURA-DEVILLECHAISE, M. SÜDHOLT. *An expressive aspect language for system applications with Arachne*, in "Transactions on Aspect-Oriented Software Development", March 2006, p. 174–213, vol. I, extended version of an article presented at AOSD'05.

[6] 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.

[7] 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, to appear.

[8] R. PIERRE, A. LÈBRE, M. CHRISTINE. *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.

[9] É. TANTER, J. FABRY, R. DOUENCE, J. NOYÉ, M. SÜDHOLT. *Expressive Scoping of Distributed Aspects*, in "AOSD '09: Proceedings of the 8th ACM international conference on Aspect-oriented software development", ACM Press, 2009, p. 27-38.

[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", vol. 72, n^{o} 1-2, 2008, p. 22–30, http://dx.doi.org/10.1016/j.scico.2007.10.005.

Year Publications

Doctoral Dissertations and Habilitation Theses

[11] H. ARBOLEDA. *FieSta: An approach for Fine-Grained Scope Definition, Configuration and Derivation of Model-Driven Software Product Lines*, Universidad de Los Andes at Bogota and Université de Nantes and École des Mines de Nantes, October 2009, Ph. D. Thesis.

[12] L. D. BENAVIDES NAVARRO. *Distributed Aspects: better separation of crosscutting concerns in distributed software systems*, Université de Nantes and École des Mines de Nantes, January 2009, Ph. D. Thesis.

[13] S. DJOKO DJOKO. *Programmation par aspects et préservation de propriétés*, Université de Nantes, June 2009, Ph. D. Thesis.

[14] F. HERMENIER. *Gestion dynamique des tâches dans les grappes, une approche à base de machines virtuelles*, Université de Nantes, November 2009, Ph. D. Thesis.

[15] M. LÉGER. *Fiabilité des Reconfigurations Dynamiques dans les Architectures à Composants*, Ecole Nationale Supérieure des Mines de Paris, May 2009, Ph. D. Thesis.

Articles in International Peer-Reviewed Journal

[16] N. ANQUETIL, U. KULESZA, R. MITSCHKE, A. MOREIRA, J.-C. ROYER, A. RUMMLER, A. SOUSA. *A Model-Driven Traceability Framework for Software Product Lines*, in "Software and Systems Modeling", 2009.

[17] P.-C. DAVID, T. LEDOUX, M. LÉGER, T. COUPAYE. *FPath and FScript: Language Support for Navigation and Reliable Reconfiguration Fractal Architectures*, in "Annals of Telecommunications, edited by Springer-Verlag France", vol. 64, n^o 1-2, January-February 2009, Special issue on Component-based architecture: the Fractal initiative.

[18] X. LEROY, H. GRALL. *Coinductive big-step operational semantics*, in "Information and Computation", vol. 207, 2009, p. 284-304.

International Peer-Reviewed Conference/Proceedings

[19] P. ANDRÉ, N. ANQUETIL, G. ARDOUREL, J.-C. ROYER, P. HNETYNKA, T. POCH, D. PETRASCU, V. PETRASCU. *JavaCompExt: Extracting Architectural Elements from Java Source Code*, in "Proceedings of the 16th Working Conference on Reverse Engineering (WCRE 2009), tool demonstration, Lille, France", October 2009, p. 317-318.

[20] H. ARBOLEDA, R. CASALLAS, J.-C. ROYER. *Dealing with Fine-Grained Configurations in Model-Driven SPLs*, in "Proceedings of the 13th International Software Product Line Conference (SPLC'09), San Francisco, CA, USA", August 2009.

[21] H. ARBOLEDA, A. ROMERO, R. CASALLAS, J.-C. ROYER. *Product Derivation in a Model-Driven Software Product Line using Decision Models*, in "Proceedings of the 12th Iberoamerican Conference on Requirements Engineering and Software Environments (IDEAS'09), Medellin, Colombia", April 2009, p. 59-72.

[22] R. DOUENCE, X. LORCA, N. LORIANT. *Lazy Composition of Representations in Java*, in "Proceedings of the 8th International Conference on Software Composition (SC'09)", LNCS, Springer Verlag, July 2009.

[23] F. FERNANDES, R. PASSAMA, J.-C. ROYER. *Event Strictness for Components with Complex Bindings*, in "ISEC'09: Proceedings of the 2nd conference on India Software Engineering Conference, New York, NY, USA", ACM Press, 2009.

[24] K. GARCÉS, F. JOUAULT, P. COINTE, J. BÉZIVIN. *Managing Model Adaptation by Precise Detection of Metamodel Changes*, in "Proc. of ECMDA 2009, Enschede, The Netherlands", Springer, June 2009.

[25] F. HERMENIER, X. LORCA, J.-M. MENAUD, G. MULLER, J. LAWALL. *Entropy: a Consolidation Manager for Clusters*, in "VEE '09: Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments, New York, NY, USA", ACM, 2009, p. 41–50.

[26] G. JÉRÔME, A. LÈBRE, V. GEOFFROY, M. CHRISTINE, G. PASCAL, S. L. SCOTT. *Refinement Proposal of the Goldberg's Theory*, in "Proceedings of the International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP'09), Taipei, Taiwan", LNCS, 2009.

[27] B. MORIN, T. LEDOUX, M. B. HASSINE, F. CHAUVEL, O. BARAIS, J.-M. JÉZÉQUEL. *Unifying Runtime Adaptation and Design Evolution*, in "IEEE 9th International Conference on Computer and Information Technology (CIT'09), Xiamen, China", October 2009.

[28] 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.

[29] A. NUÑEZ, J. NOYÉ, V. GASIUNAS. *Declarative Definition of Contexts with Polymorphic Events*, in "Proceedings of the International Workshop on Context-Oriented Programming at ECOOP'09 (COP'09), Genova, Italy", ACM Press, 2009.

[30] R. PIERRE, A. LÈBRE, M. CHRISTINE. *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.

[31] É. TANTER, J. FABRY, R. DOUENCE, J. NOYÉ, M. SÜDHOLT. *Expressive Scoping of Distributed Aspects*, in "AOSD '09: Proceedings of the 8th ACM international conference on Aspect-oriented software development", ACM Press, 2009, p. 27-38.

[32] H. N. VAN, F. D. TRAN, J.-M. MENAUD. *Autonomic virtual resource management for service hosting platforms*, in "Proceedings of the Workshop on Software Engineering Challenges in Cloud Computing, Vancouver, Canada", April 2009, p. 1–8.

[33] H. N. VAN, F. D. TRAN, J.-M. MENAUD. *SLA-Aware Virtual Resource Management for Cloud Infrastructures*, in "Proceedings of the 9th International Conference on Computer and Information Technology, Xiamen, China", vol. 1, IEEE Computer Society, 2009, p. 357-362.

National Peer-Reviewed Conference/Proceedings

[34] A. ASSAF, J. NOYÉ. *Flexible Pointcut Implementation: An Interpreted Approach*, in "Actes des journées Langages et Modèles à Objets, Nancy, France", B. CARRÉ (editor), Cépaduès-Editions, Mars 2009, p. 45-60.

[35] K. GARCÉS, F. JOUAULT, P. COINTE, J. BÉZIVIN. *A Domain Specific Language for Expressing Model Matching*, in "Actes de la 5ième édition des Journées sur l'Ingénierie Dirigée par les Modèles (IDM'09), Nancy, France", Hermes, March 2009.

[36] F. HERMENIER, A. LÈBRE, J.-M. MENAUD. *Changement de contexte pour tâches virtualisées à l'échelle de grappes*, in "Proc. of 7ème Conférence Francophone sur les Systèmes d'Exploitation (CFSE07), Toulouse", September 2009.

Scientific Books (or Scientific Book chapters)

[37] P. COINTE. *Designing Open-Ended Languages: An Historical Perspective*, in "New Computational Paradigms for Computer Music", Edition Delatour France / IRCAM Centre Georges Pompidou, 2009, p. 49–64.

Books or Proceedings Editing

[38] T. CLEENEWERCK, J. FABRY, A.-F. LEMEUR, J. NOYÉ, É. TANTER (editors). *DSAL '09: Proceedings of the 2009 AOSD workshop on Domain-specific aspect languages*, ACM Press, Charlottesville, VA, USA, 2009.

[39] M. SÜDHOLT (editor). *Proceedings of the 8th Int. Workshop on Foundations of Aspect-Oriented Languages, FOAL 2009, Charlottesville, Virginia, USA, March 2, 2009*, ACM, March 2009.

Research Reports

[40] F. HERMENIER, A. LÈBRE, J.-M. MENAUD. *Cluster-Wide Context Switch of Virtualized Jobs*, n^o RR-6929, INRIA, 2009, Research Report.

[41] G. JÉRÔME, A. LÈBRE, M. CHRISTINE. *Saline: Improving Best-Effort Job Management in Grids*, n^o RR-7055, INRIA, 2009, Research Report.

References in notes

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

[43] 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.

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

[45] 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, vol. 2192, 2001, p. 187–209.

[46] 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.

[47] 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.

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

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

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

[51] 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.

[52] F. DEREMER, H. H. KRON. *Programming-in-the-large versus programming-in-the-small*, in "IEEE Transactions on Software Engineering", vol. SE-2, n^o 2, 1976, p. 80-86.

[53] 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.

[54] 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.

[55] 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, vol. 2487, Springer-Verlag, October 2002, p. 173–188, ftp://ftp.inria.fr/INRIA/publication/publi-pdf/RR/RR-4435.pdf, preprint version is ftp://ftp.inria.fr/INRIA/publication/publi-pdf/RR/RR-4435.pdf.

[56] 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.

[57] 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, vol. 2192, Springer-Verlag, 2001, p. 170–186.

[58] 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.

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

[60] 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.

[61] A. FUGGETTA, G. P. PICCO, G. VIGNA. *Understanding Code Mobility*, in "IEEE Transactions on Software Engineering", vol. 24, n^{o} 5, May 1998, p. 342–361.

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

[63] H. GRALL. *Deux critères de sécurité pour l'exécution de code mobile*, École nationale des ponts et chaussées, CERMICS (ENPC-INRIA), 2003, Ph. D. Thesis.

[64] 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, vol. 2072, Springer-Verlag, June 2001, p. 327–353, http://www.eclipse.org/aspectj/, AspectJ web site: http://aspectj.org.

[65] 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.

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

[67] 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.

[68] 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, vol. 1616, Springer-Verlag, July 1999, p. 197–214.

[69] X. LEROY. *Coinductive Big-Step Operational Semantics*, in "ESOP", P. SESTOFT (editor), Lecture Notes in Computer Science, vol. 3924, Springer, 2006, p. 54-68.

[70] 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.

[71] 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.

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

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

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

[75] 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, vol. 1445, July 1998, p. 355-382.

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

[77] 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.

[78] D. H. NGUYEN, M. SÜDHOLT. *Property-preserving evolution of components using VPA-based aspects*, in "Proceedings of the 9th International Symposium on Distributed Objects and Applications (DOA'07).", Springer-Verlag, November 2007.

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

[80] 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.

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

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

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

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

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

[86] 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, vol. 37, 11, ACM Press, November 4–8 2002, p. 174–190.

[87] 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.

[88] 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)", vol. 26, n^{O} 5, 2004, p. 890–910.

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

[90] L. DE ALFARO, T. A. HENZINGER. *Interface Automata*, in "Proceedings of the Joint 8th European Software Engeneering Conference and 9th ACM SIGSOFT Symposium on the Foundation of Software Engeneering (ESEC/FSE-01), New York", V. GRUHN (editor), SOFTWARE ENGINEERING NOTES, vol. 26, 5, ACM Press, September 10–14 2001, p. 109–120.

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

[92] J. VAN DEN BOS, C. LAFFRA. *PROCOL: A Parallel Object Language with Protocols*, in "OOPSLA'89 Conference Proceedings: Object-Oriented Programming: Systems, Languages, and Applications", N. MEYROWITZ (editor), ACM Press, 1989, p. 95–102.