Project-Team OBASCO

OBjects, ASpects, and COmponents

Rennes - Bretagne-Atlantique

THEME COM

Activity Report

2008

Table of contents

OBASCO is a joint project between École des Mines de Nantes (EMN) and INRIA.

1. Team

Research Scientist
Jean-Marc Menaud [CR2 INRIA on leave from EMN until 31 Aug.]
Thomas Ledoux [CR1 INRIA on leave from EMN since 1 Sep. 2008]

Faculty Member
Pierre Cointe [Head of OBASCO; Professor, EMN, HdR]
Mario Südholt [Vice-head of OBASCO, designated head of ASCOLA; Associate Professor, EMN, HdR]
Rémi Douence [Associate Professor, EMN]
Hervé Grall [Associate Professor, EMN]
Jacques Noyé [designated vice-head of ASCOLA; Associate Professor, EMN]
Jean-Claude Royer [Professor, EMN, HdR]

PhD Student
Hugo F. Arboleda Jimenez [MINES & Los Andes Uni. grant (Colombia), since 1 Jan. 2006]
Ali Assaf [MESR grant, until 30 Sep. 2005]
Luis Daniel Benavides Navarro [MINES grant, until 30 Sep. 2005]
Simplice Djoko Djoko [INRIA grant shared with the project PopArt, until 30 Sep. 2005]
Fabricio de Alexandria Fernandes [CAPES grant from Brazil, since 1 Oct. 2006]
Abdelhakim Hannousse [REGIONAL COUNCIL & EMN grant, since 1 Oct.]
Fabien Hermenier [EMN grant, since 1 Oct. 2006]
Kelly Garcés [ANR FLFS & EMN grant, since 1 Oct. 2007]
Mayleen Lacouture [REGIONAL COUNCIL & IST AMPLE grant, since 1 Oct.]
Dong Ha Nguyen [REGIONAL COUNCIL & AOSD-Europe grant, until 15 Apr.]
Angel Núñez [MINES & IST AMPLE grant, since 1 Oct. 2006]

Post-Doctoral Fellow
Nicolas Loriant [ANR SELFWARE grant until 31 Aug.]
Pierre-Charles David [ANR SELFWARE grant until 29 Feb.]
Joost Noppen [IST AMPLE grant until 31 Aug.]
Nicolas Anquetil [IST AMPLE grant since 1 Feb.]

Visiting Scientist
Awais Rashid [Lancaster University, from 5 May to 23 Jul.]

Administrative Assistant
Diana Gaudin [Part-time (50%)]
Élodie Lizé [Part-time (50%)]

2. Overall Objectives

2.1. Evolution of the overall objectives: the project-team ASCOLA

The overall objectives of the OBASCO project-team have been revised in 2008 resulting in the definition of a new project-team ASCOLA in March 2008, whose evaluation phase ended in August. As of Nov. 2008, ASCOLA is awaiting its formal creation notice.

The presentation of the overall objectives, foundations and results in this activity report have been structured according to the definition of the ASCOLA project-team.

2.2. Presentation

The OBASCO project-team team aims at harnessing and developing advanced application structuring mechanisms, and supporting the transformational development of correct large-scale applications as well as their valid evolution throughout their entire life cycles. 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 (that enable sophisticated relationships between execution events to be formulated and manipulated directly at the language level) to modularize crosscutting concerns. 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.

2.3. Highlights of the year

Two results of OBASCO deserve particular notice:

- We have set up and will coordinate the ANR project “SelfXL” that involves three academic and two industrial partners on the topic of self-adaptive, autonomic, large-scale, heterogeneous distributed applications (for details see Sec. 8.2).

  This project constitutes a major milestone for the transition from OBASCO project-team to its successor ASCOLA in that it involves the development of new domain-specific languages, techniques for component reconfiguration and aspect-oriented languages for large-scale distributed systems, in particular, clusters.
- We have also been able to solve, jointly with researchers from VU Brussels, a fundamental problem of AspectJ-like aspect languages that has persisted since the inception of aspect-oriented programming in 1996. We have defined the first static type system for the basic mechanisms of such languages, in particular including `proceed`, that is equipped with a type soundness proof. A corresponding publication has received a distinguished paper award at the international conference AOSD’08 (for details see Sec. 6.1).

3. Scientific Foundations

3.1. Overview

Since we mainly work on new software structuring concepts and programming language design, we first briefly introduce some basic notions and problems of software components (understood in a broad sense, i.e., including modules, objects, 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 [75]. At that time, the scope of a component was limited to a single procedure. In the seventies, the growing complexity of software made it necessary to consider a new level of structuring and programming and led to the notions of information hiding, *modules*, and module interconnection languages [85], [57]. Information hiding promotes a black-box model of program development whereby a module implementation, basically a collection of procedures, is strongly encapsulated behind an interface. This makes it possible to guarantee logical invariant *properties* of the data managed by the procedures and, more generally, makes *modular reasoning* possible. In a first step, it is possible to reason locally, about the consistency between the module implementation and the module interface. In a second step, it is possible to reason about composing modules by only considering their interfaces. Modern module systems also consider types as module elements and consider, typically static, modules as a unit of separate compilation, with the most sophisticated ones also supporting modules parametrized by modules [74].

Object-Oriented Programming. *Classes* and *objects* provide another kind of software component, which makes it necessary to distinguish between *component types* (classes) and *component instances* (objects). Indeed, unlike modules, objects can be created dynamically. Although it is also possible to talk about classes in terms of interfaces and implementations, the encapsulation provided by classes is not as strong as the one provided by modules. This is because, through the use of inheritance, object-oriented languages put the emphasis on *incremental programming* to the detriment of modular programming. This introduces a white-box model of software development and more flexibility is traded for safety as demonstrated by the *fragile base class* issue [79].

Architecture Description Languages. The advent of distributed applications made it necessary to consider more sophisticated connections between the various building blocks of a system. The *software architecture* [88] of a software system describes the system as a composition of *components* and *connectors*, where the connectors capture the *interaction protocols* between the components [48]. It also describes the rationale behind such a given architecture, linking the properties required from the system to its implementation. *Architecture Descriptions Languages* (ADLs) are languages that support architecture-based development [76]. A number of these languages make it possible to generate executable systems from architectural descriptions provided implementations for the primitive components are available. However, guaranteeing that the implementation conforms to the architecture is an issue.

3.3. Aspect-Oriented Programming

The main driving force for the structuring means, such as components and modules, is the quest for clean *separation of concerns* [59] on the architectural and programming levels. It has, however, early been noted that concern separation in the presence of crosscutting functionalities requires specific language and implementation level support. Techniques of so-called *computational reflection*, for instance, Smith's 3-Lisp or Kiczales's CLOS meta-object protocol [89], [71] as well as metaprogramming techniques have been developed to cope with this problem but proven unwieldy to use and not amenable to formalization and property analysis due to their generality.

Aspect-Oriented Software Development [70], [46] has emerged over the previous decade as the domain of systematic exploration of crosscutting concerns and corresponding support throughout the software development process. The corresponding research efforts have resulted, in particular, in the recognition of *crosscutting* as a fundamental problem of virtually any large-scale application, and the definition and implementation of a large number of aspect-oriented models and languages.

However, most current aspect-oriented models, notably AspectJ [69], rely on pointcuts and advice defined in terms of individual execution events. These models are subject to serious limitations concerning the modularization of crosscutting functionalities in distributed applications, the integration of aspects with other modularization mechanisms such as components, and the provision of correctness guarantees of the resulting AO applications. They do, in particular, only permit the manipulation of distributed applications on a per-host basis, that is, without direct expression of coordination properties relating different distributed entities [90]. Similarly, current approaches for the integration of aspects and (distributed) components do not directly express interaction properties between sets of components but rather seemingly unrelated modifications to individual components [56]. Finally, current formalizations of such aspect models are formulated in terms of low-level semantic abstractions (see, e.g., Wand's et al semantics for AspectJ [92]) and provide only limited support for the analysis of fundamental aspect properties.

Recently, first approaches have been put forward to tackle these problems, in particular, in the context of so-called *stateful* or *history-based aspect languages* [60], [61], which provide pointcut and advice languages that directly express rich relationships between execution events. Such languages have been proposed to directly express coordination and synchronization issues of distributed and concurrent applications [84], [50], [63], provide more concise formal semantics for aspects and enable analysis of their properties [49], [62], [60], [47]. 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 [96], [83], component interactions [93], [86], [94] and service orchestrations [58] are most frequently expressed in terms of *regular interaction protocols* that enable basic properties, such as compatibility, substitutability, and deadlocks between components to be defined in terms of basic operations and closure properties of finite-state automata. Furthermore, such properties may be analyzed automatically using, e.g., model checking techniques [54], [65].

However, the limited expressive power of regular languages have led to a number of approaches using more expressive *non-regular* interaction protocols that typically provide context-free or turing-complete expressiveness [87], [53]. While these protocol types allow conformance between components to be defined (e.g., using unbounded counters), property verification can only be performed manually or semi-automatically.

Furthermore, first approaches for the definition of *aspects over protocols* have been proposed, as well as over regular structures [60] and non-regular ones [91], [81]. 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 [82].

3.5. Patterns

Patterns provide a kind of abstraction that is complementary to the modularization mechanisms discussed above. They have been used, in particular, to define general *architectural styles* either by defining entire computation and communication topologies [80], connectors between (complex) software artifacts [77], or (based on, possibly concretizations of, *design patterns* [68]) as building blocks for object-oriented software architectures. The resulting pattern-based architectures are similar to common component-based architectures and are frequently used to implement the latter, see, for instance, Sun's J2EE patterns.

Patterns have also been used to implement architectural abstractions. This is the case, for instance, for the numerous variants of the *publish/subscribe pattern* [64] as well as the large set of so-called *skeletons* [55], that is, patterns for the implementation of distributed and concurrent systems. While these patterns are essentially similar to architecture-level patterns, their fine-grained application to multiple code entities often results in crosscutting code structures.

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 represent domain knowledge in terms of suitable basic language constructs and their compositions at the language level. By trading generality for abstraction, they enable complex relationships among domain concepts to be expressed concisely and their properties to be expressed and formally analyzed. DSLs have been applied to a large number of domains; they have been particularly popular in the domain of software generation and maintenance [78], [95].

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

Underlying our interest in these domains is the well-known observation that large-scale distributed applications are subject to *numerous crosscutting functionalities* (such as the transactional behavior in enterprise information systems, the implementation of security policies, and fault recovery strategies). These functionalities are typically partially encapsulated in distributed infrastructures and partially handled in an ad hoc manner by using infrastructure services at the application level. Support for a more principled approach to the development and evolution of distributed software systems in the presence of crosscutting functionalities has been investigated in the field of *open adaptable middleware* [52], [73]. Open middleware design exploits the concept of reflection to provide the desired level of configurability and openness. However, these approaches are subject to several fundamental problems. One important problem is their insufficient, framework-based support that only allows partial modularization of crosscutting functionalities.

There has been some *criticism* on the use of *AspectJ-like aspect models* (which middleware aspect models like that of JBoss AOP are an instance of) for the modularization of distribution and concurrency related concerns, in particular, for transaction concerns [72] and the modularization of the distribution concern itself [90]. 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 OBASCO 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 [67] 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 [49], [63].

4. Application Domains

4.1. Enterprise Information Systems

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

We have developed new techniques to debug and test such infrastructures, see Sec. 6.2, and implemented these techniques in the context of the AWED system for explicitly distributed aspect-oriented programming, see Sec. 5.2. 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 Architectures (SOAs) have been proposed as a solution to some of the integration problems discussed above and are gaining importance as a major architecture, among others, because of their loosely-coupled model of service assembly and formally defined protocols (e.g., BPEL workflows). However, SOAs are subject to several important limitations from a software engineering viewpoint. First, the (regular) protocols currently employed support few correctness guarantees. Quality of service properties that have to be respected by service composition are typically not taken into account. More expressive protocols to define contracts between services [51] promise to provide stronger correctness guarantees. Second, similar to industrial component models, such as EJBs, SOAs currently provide only limited support for crosscutting services, such as transaction and security services.

This year we have provided new results on QoS properties of SOAs, see Sec. 6.2 and improved our understanding of using symbolic transition systems for non-regular protocols, see Sec. 6.1.

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. The complexity and the growing need of dynamic evolution of such 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 applied respectively to (clustered) J2EE applications servers and scientific (grid) applications.

This year we have improved on existing task placement algorithms for clusters and applied distributed aspects to the evolution of grid algorithms, see Sec. 6.5. We have developed new use cases dealing with self-administration of clusters with the partners of the Selfware project, see Sec. 8.2. Furthermore, we have set up and will coordinate the SelfXL national project, see Sec. 8.2, on the evolution and reconfiguration of large-scale clusters.

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 we have extended the model of concurrent aspects to the explicit modelling of contexts. This facilitates reasoning on context-aware systems and improves their modularity, see Sec. 6.5.

5. Software

5.1. Arachne

Keywords: *AOP*, *C language*, *dynamic system evolution*, *proxies*.

Participants: Jean-Marc Menaud [correspondent], Rémi Douence, Mario Südholt, Nicolas Loriant.

We have implemented the EAOP model into Arachne, an aspect dynamic weaver for C legacy applications. Arachne has been developed to dynamically evolve a system at runtime without causing service interruptions, for instance, in order to modify prefetching policies in Web caches or security updates in proxies. C applications, in particular those using operating system level services, frequently comprise multiple crosscutting concerns: network protocols and security are typical examples of such concerns. While these concerns can partially be addressed during design and implementation of an application, they frequently become an issue at runtime, e.g., to avoid server downtime when discovering a buffer overflow that implies critical breaches in the security model of an application. These concerns are crosscutting in the sense of AOP and aspects are therefore a means of choice for their modularization.

The Arachne framework is built around two tools, an aspect compiler and a runtime weaver based on new hooking strategies. Arachne implements weaving by exploiting linking information to rewrite C binary executables on the fly. With this approach, we can extend the base program using AOP without loss of efficiency and without service interruption. Furthermore, Arachne does not need any preparation of the base program to enable aspect weaving.

5.2. Awed

Keywords: *AOP*, *distributed programming*, *invasive patterns*.

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 we have extended the AWED language and implementation by causal patterns, that is, sequence aspects whose ordering may be determined in terms of logical clocks. This enables programmers to refer to temporal orderings of events and reorder events according to such orderings. We have applied this feature to two Java-based middleware platforms, see Sec. 3.7. Furthermore, AWED has been used to define *invasive distributed patterns*, a new notion of patterns supporting the compositional construction of large-scale distributed applications. In particular, we have shown how invasive patterns can be used to dynamically evolve grid-based algorithms using AWED, see Sec. 6.5. Finally, AWED has been used in the European network of excellence AOSD-Europe and in the project AMPLE (see Sec. 8.3).

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

5.3. FPath and FScript

Keywords: *Fractal*, *autonomic computing*, *dynamic reliable reconfiguration*, *self-adaptive components*.

Participants: Thomas Ledoux [correspondent], Pierre-Charles David, Marc Léger, Mayleen Lacouture.

FPath and FScript are two Domain-Specific Languages (DSLs) dealing with respectively the navigation and the dynamic reconfigurations of Fractal architectures [13]. *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.

This year, we have extended FPath to the JMX architectures [44]. The objective of this extension is to offer the language support provided by FPath in systems already using JMX as management platform. In that way, administrators are able to navigate using FPath notation through MBean servers to monitor the application resources. Furthermore, FScript is used by the projects Selfware and Galaxy (see Sec. 8.2).

FScript and its extensions are downloadable under LGPL at http://fractal.objectweb.org/fscript.

5.4. STSLib

Keywords: *Architecture*, *Symbolic Transition Systems*, *component*, *configuration graph*, *synchronous product*.

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 consise 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, behavioural compatibility, and *event strictness* checking. A composite is event strict iff each subcomponent is event strict and if each communication defined at the composite level occurs in the dynamic behaviour of the composite. It is intended to provide tool support for designing applications based on STSs and experimenting new constructions for components and architectures.

5.5. WildCAT

Keywords: *complex event processing*, *context-aware applications*, *monitoring*.

Participants: Thomas Ledoux [correspondent], Nicolas Loriant.

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 interrogate the resources: a pull mode relies on synchronous communication and a push one relies on asynchronous communication. WildCAT supports remote communication via JMS (Java Message Service) and RMI (Remote Method Invocation).

WildCAT is an open source component used by the projects Selfware and Galaxy (see Sec. 8.2). WildCAT is downloadable under GPL v2 at http://wildcat.objectweb.org.

6. New Results

6.1. Components and Aspects

Participants: Ali Assaf, Fabricio Fernandes, Jacques Noyé, Jean-Claude Royer, Mario Südholt, Pierre-Charles David.

OBASCO has achieved different results on the foundations and implementation of aspect-oriented and component-based systems. We have contributed to the foundations of AOP by proposing the first safe type system for AspectJ-like languages that closes fundamental holes in the current type system of AspectJ. We have clarified the relationship between dynamic and static implementation strategies of aspect languages by investigating a hybrid model and implementation for their translation and execution. As to software components, we have deepened our understanding of compositional systems built using (turing-complete) protocols formulated using symbolic transition systems. We have provided, in particular, new results on their theoretical properties and developed new implementation techniques. Finally, we have provided new means of how to integrate rule-based systems with software components.

6.1.1. Type Safety for AspectJ-like Languages

We have been able to solve, jointly with researchers from VU Brussels, a fundamental problem concerning aspect languages that follow the AspectJ model. In fact, since the inception of AOSD and the earliest versions of AspectJ, such languages have been subject to important typing problems. While the static type discipline of Java is only subject to a small hole concerning type casts, object-oriented AspectJ-like languages have been and are subject to large holes in their typing discipline. The application of advice to join points of a base program cannot, in general, be typed correctly if around advice with the pseudo-method `proceed` is used. Previous approaches to this problem only addressed it partially.

The main technical difference of our approach to its predecessors is that the typing of join points, pointcuts and advice (including `proceed`) is based on *type ranges*. This extension makes it possible to correctly type applications of `proceed`. In order to do so, two interfaces have to be provided: one towards the base joinpoint and another one to the advice (that calls `proceed`). These two interfaces require that the argument types and the result types are sometimes bounded from below sometimes from above, hence the type ranges. We have formally defined the corresponding aspect and base languages, a type system, and a dynamic semantics. Based on this definition, we have been able to give the first type soundness proof including all basic mechanisms of AspectJ-like languages, in particular `proceed`.

The motivation of the typing problems, the design choices of the type system, and essential parts of its formalization including a sketch of the soundness proof have been published at AOSD'08 and received one of two distinguished paper awards [27]. The complete formalization can be found in a technical report [66]. Finally, two prototype implementations of the type system are available: one as an extension of the AspectJ compiler implemented using the compiler system abc, a second one as a Java framework based on (see http://ssel.vub.ac.be/ssel/research/aosd/strongaspectj).

6.1.2. Dynamic AspectJ

We have considered the difficulties linked to the static scheduling strategy of AspectJ and have shown how to overcome them by turning to a more dynamic strategy, making it possible to order, cancel, and deploy aspects at runtime. This more dynamic strategy can be obtained by a minor update of the semantics of AspectJ introducing the notion of current *aspect group*, that is, the aspects scheduled for the current join point. We have shown how to reflect this change at the language level and presented a prototype of the resulting AspectJ variant, *Dynamic AspectJ* [20]. This prototype reuses AspectJ to perform a first step of static weaving, which we complement by a second step of dynamic weaving, implemented through a thin interpretation layer. This can be seen as an interesting example of reconciling compilers and interpreters, the static and the dynamic world. This work is actually part of a larger framework that we call CALI, for *Common Aspect Language Interpreter*. CALI is used to prototype aspect languages and study multi-aspect language AOP (see http://www.emn.fr/x-info/cali). In [37], we have revisited the semantics of the call and execution pointcuts of AspectJ, which are used to select the invocation execution points of interest from the caller and the callee standpoint, respectively. We have shown how existing alternative semantics, including a dynamic semantics, which had not been implemented before, can easily be implemented and experimented with using CALI.

6.1.3. Component Models with Interface Protocols

In his thesis, Sebastian Pavel [12] studied the possibility of equipping a simple component model with explicit protocols based on STSs. In order to do so, he proposed a process calculus for STSs, based on the Finite State Processes of Magee and Kramer. This calculus was used as a direct model for implementing a concrete prototype of the component model. For the sake of simplicity, binary communications only were considered. Also, for the sake of efficiency, *mixed states*, which occur in the presence of choices mixing inputs and outputs and require to put in place a heavy handshake mechanism, were excluded. This made it possible to provide an efficient distributed implementation of the system. An alternative infrastructure, based on a centralized arbiter, was also studied. Such a scheme potentially supports both n-ary communications and mixed states. Based on this work, we designed a software library [45] to support n-ary communications and a more complex synchronisation mechanism.

The STSLib project (see Sec. 5.4) aims at defining an environment to formally specify and execute software components. One important feature is that our components are equipped with symbolic transition systems that glue together a protocol with guards, input/output notations, and data types. These sophisticated protocols are well-suited to the design of concurrent and communicating systems but verification remains a difficult challenge. We expect to narrow the gap between the design level and the programming level by providing an effective runtime support for STSs. In [31], we state the main objectives of the STSLib project and review its current state. We address the formal description of a component model, a specific approach to verify these systems, and a survey of the operational infrastructure necessary for their execution. This work shows that an automatic Java translation of the data part description is possible.

The STSLib was experimented on several small and middle size examples and we observed that N-party communication has some benefits over binary communication. In order to improve software modularity promoted by component-based software engineering, we have introduce three new types of binding in STSLib. We have formalized our model and defined two properties: behavioural compatibility and event strictness. This last notion is orthogonal to behavioural compatibility. Its purpose is to check if a communication designed by an architect really occurs in the global system behaviour. Contrary to behavioural compatibility we explicit a static checking in [30].

6.1.4. Programming Component-Based Systems with Active Rules

Several proposals have tackled architectural issues involved in the implementation of a control loop for autonomic computing but none of them are based on an explicit and extensive use of component programming for implementing the autonomic features themselves. In [33], we have proposed an architecture for the integration of active rules into component-based systems in which the rules as well as their semantics (execution model, behaviour) are implemented as components. This permits i) to construct personalized rule-based systems and ii) to dynamically modify them and their semantics in the same manner as the underlying system.

6.2. Large-Scale Distributed and Concurrent Systems

Participants: Daniel Benavides, Rémi Douence, Thomas Ledoux, Noyé Jacques, Mario Südholt.

OBASCO has worked on three types of results for large-scale distributed systems. First, we have developed new means for the definition of quality of service (QoS) policies and their execution over standard infrastructures for service-oriented architectures. Second, we have extended our approach of distributed aspect-oriented programming (AWED) with support for event sequences that are subject to logical time constraints. This work has yielded interested applications, in particular, for the debugging and testing of middleware infrastructures. Finally, we have proposed new means for the precise definition and manipulation of distributed applications based on a notion of the scope of distributed aspects expressed in terms of fine-grained distributed event sequences.

6.2.1. QoS Policies for Business Processes in Service-Oriented Architectures

The advent of Service-Oriented Architectures tends to promote a new kind of software architecture where services, exposing features accessible through highly standardized protocols, are composed in a loosely-coupled way. In such a context, where services are likely to be replaced or used by a large number of clients, the notion of Quality of Service (QoS), which focuses on the quality of the relationship between a service and its customers, becomes a key challenge.

To cope with this issue, we propose to ease QoS management in service compositions through a better separation of concerns [21], [11]. For this purpose, we designed QoSL4BP, a domain-specific language which allows QoS policies specification for business processes. More specifically, the QoSL4BP language is designed to allow an architect to specify QoS constraints and mechanisms over parts of BPEL compositions. This language is executed by our ORQOS platform, which cooperates in a non-intrusive way with orchestration engines. At pre-deployment time, the ORQOS platform performs service planning depending on the QoS offers and requirements of each service expressed in the QoSL4BP policies. At runtime, QoSL4BP policies make it possible to react to QoS variations and to enact QoS management related mechanisms.

6.2.2. Causal Patterns for Debugging and Testing of Middleware

Many tasks that involve the dynamic manipulation of middleware and large-scale distributed applications, such as debugging and testing, require the monitoring of intricate relationships of execution events that trigger modifications to the executing system. Furthermore, events often are of interest only if they occur as part of specific execution traces, not in all possible non-deterministic interleavings of events in these traces. Current techniques and tools for the definition of such manipulations provide only very limited support for such event relationships and do not allow concise definitions of restrictions on the interleaving of events.

We have extended our model for distributed aspects by notions of causal relations, thus enabling pointcuts to predicate over the order of execution events and enabling advice to reorder messages sent in a distributed system [24]. We have argued for the use of aspect-based high-level programming abstractions for the definition of relationships between execution events of distributed systems and the control of non-deterministic interleavings of events. Concretely, we have provided the following contributions: we (i) have motivated that such abstractions improve on current debugging and testing methods for middleware, (ii) introduced corresponding language support for pointcuts and advice defined in terms of causal event sequences, and (iii) evaluated our approach in the context of the debugging and testing of two different Java-based middleware infrastructures, JBoss Cache for replicated caching and the Apache ActiveMQ message broker.

6.2.3. Aspect Scoping for Distributed Applications

We have, in cooperation with researchers from the University of Chile, developed new techniques for the definition and control of aspect scoping in distributed applications. These techniques allow aspects to be propagated along execution paths of a distributed base application. They support, in particular, a very fine-grained notion of distributed control flow [36].

6.3. Property-Centric Definition of Aspects

Keywords: *formal semantics and properties*, *model checking*, *proof*, *sequential and distributed aspects*.

Participants: Daniel Benavides, Simplice Djoko Djoko, Rémi Douence, Mario Südholt.

OBASCO 's work on the foundations of AOP has been focused on two fundamental issues in 2008. First, we have extended previous approaches that categorizes aspects with respect to the semantic properties they satisfy. We have been able, in particular, to characterize such classes precisely and constructively in terms of a corresponding set of formal languages. Second, we have sketched a first formal semantics of distributed aspects in terms of labelled transition systems that are amenable to property verification using model checking.

6.3.1. Semantics of Sequential Aspect Languages

Aspect-Oriented Programming can, in general, arbitrarily modify the semantics of programs to which aspects are applied. In particular, weaving can invalidate crucial safety and liveness properties of the base program. We have identified categories of aspects that preserve some classes of properties. It is then sufficient to check that an aspect belongs to a specific category in order to ensure that the corresponding properties will be preserved through weaving. These categories are defined precisely based on a language-independent abstract semantics framework. The classes of properties are defined as subsets of the temporal logic LTL for deterministic programs and CTL* for non-deterministic ones. We have formally proved that, for any program, the weaving of any aspect in a category preserves any property in the corresponding class [28].

We have then explored the problem of how to provide programmers with simple means to express the corresponding categories. To this end, we have, for each aspect category, designed a specialized aspect language which ensures that any aspect written in that language belongs to the corresponding category. The aspect languages share the same expressive pointcut language and are designed with respect to a common imperative base language. It can be proved that these languages preserve the corresponding classes of properties by construction. We have proved, for instance, that all aspects written in the observer language belong to the corresponding category [29].

6.3.2. Semantics of Distributed Aspects

As part of the development of the AWED approach to distributed aspects (see Sec. 5.2) we have defined invasive patterns, an extension of standard parallel and distributed architectural patterns for complex distributed algorithms. These patterns have been implemented in terms of the AWED language, an aspect language with features for explicit distribution. In order to explore the semantics properties of distributed aspects, we have defined two (partial) formal semantics [23] based on labeled transition systems, one for AWED and the other for invasive patterns. These semantics that have been defined based on the process calculus FSP. Based on these semantics, first results have been obtained on liveness and safety properties of distributed aspects and invasive patterns. The properties have been checked using model checking techniques.

6.4. A domain-specific approach to software development

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

We have pursued our work on the use of DSLs as a general tool for software development in three different dimensions. First, we have abstracted the Fractal DSL for the reconfiguration of Fractal components into a DSL that allow reconfigurations to be defined for a set of, partially very different, component models. Second, we have developed our previous results on a reflective approach to the definitions of DSLs and aspect languages. Finally, we have started to explore the transformation of models depending on evolving metamodels.

6.4.1. A Multi-Dimensional Evolution of FScript: Towards more Genericity and Reliability

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 make it complex to write reconfigurations and ensure the reliability of these reconfigurations. We proposed FScript, a Domain-Specific Language (DSL), to solve these issues: direct and focused language support for architecture navigation and reconfiguration make it easier both to write the reconfigurations and to guarantee several properties by construction (e.g., termination of scripts by excluding the possibility of infinite loops) [13].

The DSL FScript (see Sec. 5.3) was the starting point of two different research axes: (i) a multi-stage approach to ensure reliable dynamic reconfigurations; (ii) a model-driven engineering approach to provide a generic version of FScript.

In the first project, we elaborate an end-to-end solution to define and execute reliable dynamic reconfigurations of open component-based systems while guaranteeing their continuity of service [26]. This solution relies on a multi-stage approach in order to deal with the different kinds of possible errors in the most appropriate way; in particular, the goal is to detect errors as early as possible to minimize their impact on the target system. Reconfigurations are expressed in FScript in order to allow different levels of static and dynamic validation, thus detecting errors before executing the reconfiguration where possible. For errors that can not be detected early (including software and hardware faults), a runtime environment provides transactional semantics to the reconfigurations.

In the second project, we propose a generic reconfiguration language, and an implementation method, using model-driven engineering tools, which allows a semi-automatic instantiation of the reconfiguration language for each target framework (e.g., JMX, OSGi, SCA). Starting from FScript, dedicated to Fractal applications, the generic reconfiguration language formalizes some abstract and common concepts allowing the architectures in any framework to be described. A key to express architectural consistency is the concept of constraints, which define invariant properties to be preserved. The implementation method consists of defining a domain-specific language to translate any elementary reconfiguration action into calls to the reconfiguration API of the framework [44].

Finally, we are currently investigating a kind of merge of these two related projects to design a generic and reliable dynamic reconfiguration language.

6.4.2. Support for the Implementation and Composition of Domain-Specific Aspect Languages

Reflex was one of the first infrastructures to support both the implementation and the composition of Domain-Specific Aspect Languages (DSALs [25]). In [14], we briefly overview the foundations of Reflex (metaprogramming, computational reflection, and aspect-oriented programming) and present the infrastructure. Based on a uniform model of partial reflection, Reflex provides both structural and behavioral facilities, which makes it easy to experiment with (combinations of) advanced uses of AOP and reflection without reinventing the wheel or being limited to a specific AOP language.

6.4.3. Adaptation of Models to Evolving Metamodels

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 software evolution.

A first result described in [42] was to deal with the issue of automatic adaptation of models to their evolving metamodels in the field of Model-Driven Engineering. This common work done jointly with the AtlanMod team proposes a general two-step solution. First we compute equivalences and differences between the metamodels and expresses these into a so called *weaving model*. Then we translate this model into an executable transformation. The technical report illustrates this solution, using the AMMA platform and its MM3 DSL on two concrete scenarios dealing with Petri nets and Java Netbeans, respectively. The next step will be the definition of a DSL to express (meta)models matching and to compute the corresponding weaving models.

6.5. Application Domains

Participants: Jean-Marc Menaud, Fabien Hermenier, Mario Südholt, Daniel Benavides, Rémi Douence, Jacques Noyé, Angel Núñez.

OBASCO has achieved application-oriented results in three different domains. (These result complement those involving application-oriented validation of mainly research-oriented results, in particular in the domains of software-oriented architectures and middleware for enterprise information systems, presented in the previous sections.) First, we have proposed a novel approach to the dynamic reconfiguration of tasks in clusters. Second, we have applied invasive distributed patterns to the dynamic evolution of grid applications. Finally, we have shown how to better modularize context-aware applications using a notion of concurrent aspects.

6.5.1. Task Placement within Clusters

Clusters provide powerful computing environments, but in practice much of this power goes to waste, due to the static allocation of tasks to nodes, regardless of their changing computational requirements. Dynamic consolidation is an approach that migrates tasks within a cluster as their computational requirements change, both to reduce the number of nodes that need to be active and to eliminate temporary overload situations. Previous consolidation strategies have relied on task placement heuristics that use only local optimization and typically do not take migration overhead into account. However, heuristics based on only local optimization may miss the globally optimal solution, resulting in unnecessary resource usage, and the overhead for migration may nullify the benefits of consolidation.

We propose the Entropy resource manager for homogeneous clusters. Entropy performs consolidation based on constraint programming and takes migration overhead into account. The use of constraint programming allows Entropy to find mappings of tasks to nodes that are better than those found by heuristics based on local optimizations, and that are frequently globally optimal in the number of nodes [38]. Because migration overhead is taken into account, Entropy chooses migrations that can be implemented efficiently, incurring a low performance overhead [32].

6.5.2. Evolution of Grid Algorithms using Invasive Patterns

The development of grid algorithms is frequently hampered by limited means to describe topologies and lack of support for the invasive composition of legacy components in order to pass data between them. Topologies are often defined in an ad hoc and low-level manner, e.g., defining sets of vertices and edges. Invasive access to grid component implementations is frequently necessary to enhance them, for instance, by fault tolerance mechanisms.

We have proposed a solution [22] to these issues based on the notion of invasive patterns for the construction of distributed algorithms, an extension of well-known computation and communication patterns that we introduced in 2007. Based on a study of how patterns are instantiated in NAS Grid, a well-known benchmark used for evaluating performance of computational grids, we show how invasive patterns can be used for the declarative definition of large-scale grid topologies and checkpointing algorithms.

6.5.3. Context-Aware Computing

Context-aware applications adapt their behaviour depending on changes in their environment context. Programming such applications in a modular way requires to modularize the global context into more specific contexts and attach specific behaviour to these contexts. This is reminiscent of aspects and has led to the notion of context-aware aspects. We have revisited this notion of context-aware aspects in the light of previous work on concurrent event-based aspect-oriented programming (CEAOP) and shown how CEAOP can be extended in a seamless way in order to define a model for the coordination of concurrent adaptation rules with explicit contexts [34]. This makes it possible to reason about the compositions of such rules. The model is concretized into a prototypical modeling language, available at http://www.emn.fr/x-info/anunezlo/ltsa-modeling/.

7. Contracts and Grants with Industry

7.1. France Telecom R&D PhD about Web Services

Participants: Thomas Ledoux, Fabien Baligand.

Web Services and Service-Oriented Architectures aim to deliver agile service infrastructures. While the language BPEL has emerged to allow the specification of Web Services compositions from a functional point of view, it is still left to the architects to find proper means to handle the Quality of Service (QoS) concerns of their compositions.

Fabien Baligand's PhD work [11] overcomes this shortcoming by introducing both a new language and a platform for QoS specification and implementation in service compositions. More specifically, we propose a policy-based language aiming to provide expressive QoS behavioural logic specifications (e.g., QoS constraints processing, QoS mechanisms injection) for Web Service orchestrations, as well as a non-intrusive platform in charge of their execution both at pre-deployment time and at runtime [21] .

This work has been supported by France Telecom R&D (MAPS/AMS) for an amount of 22.5 KEUR.

7.2. France Telecom R&D PhD about Fractal Architectures

Participants: Thomas Ledoux, Marc Léger.

Reliability is a main problem in systems subject to dynamic reconfigurations. Marc Leger's PhD work ensures the reliability of dynamic sequential, concurrent and distributed reconfigurations in the Fractal component model. Furthermore, his model can be generalized to other component models. We have shown how ACID properties in the context of reconfigurations can improve reliability by making systems fault-tolerant, i.e., compliant with the specification in spite of faults due to dynamic reconfigurations. The results has been used in the Selfware national project (see Sec. 8.2) as a self-healing foundation of autonomic distributed applications.

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

7.3. 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 clients' expected request rate), 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* applies model engineering, aspect-oriented programming, and domain specific languages to the field of real-time systems. This is joint work between teams from the LINA and IRCCyN institutes: AtlanMod, COLOSS, MODAL, and STR. See also the FLFS ANR related project in Sec. 8.2.

Mario Südholt co-coordinates the cluster and OBASCO 's funding amounts to 53 KEUR including half a PhD thesis grant.

8.2. National Projects

8.2.1. ANR/RNTL Selfware

Participants: Thomas Ledoux, Jean-Marc Menaud, Pierre-Charles David, Nicolas Loriant.

The Selfware project (http://sardes.inrialpes.fr/selfware) is an ANR/RNTL project running for 30 months which has been submitted and accepted in 2005 for funding amounting to 222 KEUR from June 2006 on.

The goal of Selfware is to propose a software infrastructure enabling the building of distributed applications under *autonomic* administration. Historically, Autonomic Computing is an initiative started by IBM Research in 2001, where the word *autonomic* is borrowed from physiology; as a human body knows when it needs to breathe, software is being developed to enable a computer system to know when it needs to repair itself, configure itself, and so on.

In the Selfware project, we are interested in the autonomic administration of computing systems that involve the following characteristics: self-configuring, self-healing and self-optimizing of distributed applications. We focus on two types of server administration: (i) J2EE application servers with Jonas; (ii) asynchronous Message-Oriented Middleware with Joram.

Experiments are realized in the OW2 context (http://www.ow2.org) with the Fractal component model. The main contributions of our team are the DSL FScript (see Sec. 5.3) and the monitoring service WildCAT (see Sec. 5.5).

The project federates work between six partners: France Telecom R&D, Bull, Scalagent, INRIA Rhône-Alpes (the SARDES project-team), IRIT-ENSEEIHT, and OBASCO .

8.2.2. ANR non thématique 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 is 36 months, starting December 2006. The OBASCO funding part amounts to 70 KEUR. The Web page is :http://flfs.emn.fr).

8.2.3. ANR/ARPEGE SelfXL

Participants: Jean-Marc Menaud, Thomas Ledoux.

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 from January 2009 on.

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

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

We propose to design a decision module based on 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, Jean-Marc Menaud.

The Galaxy (http://galaxy.gforge.inria.fr/) ADT (Technology Development Action) intends to strengthen INRIA's position in the domain of SOA. This ADT aims at pre-assembling technological bricks from various teams, projects and preparing them to be transferred through the open source software channel.

The goal that the Galaxy ADT aims at achieving is to provide an IT agile platform, built on dynamic software architecture principles, and fitting for flexibility, dynamical reconfiguration, adaptability, continuity and autonomic computing. Fractal, SCA-Tinfi and GCM-ProActive are the major technologies which will be the technological drivers of this ADT. The different usage scenarios will demonstrate that this platform is able to support the design, modelling, deployment and execution of business processes. At the same time, the ADT will target the definition of a new common language to manipulate dynamically adaptive distributed SOA-based systems, encompassing application and middleware layers.

From an engineering point of view, the OBASCO 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 the new common language to manipulate dynamically adaptive distributed SOA-based systems.

Contributors to this ADT are mainly research project-teams, including ADAM, ECOO, OASIS, OBASCO , OW@INRIA, SARDES and TRISKELL.

The duration of this ADT is over 28 months : kickoff has been held on July 3rd, 2008 and the project is planned to end in October 2010. The galaxy ADT is led and managed by the OW@INRIA team.

8.3. European Projects

8.3.1. NoE AOSD-Europe

Participants: Pierre Cointe, Rémi Douence, Jacques Noyé, Mario Südholt, Ali Assaf, Daniel Benavides, Simplice Djoko Djoko.

OBASCO participated in the European Network of Excellence in Aspect-Oriented Software Development (NoE AOSD) from September 2004 to November 2008. This network federated the essential part of the European research community in AOSD. The network was coordinated by Lancaster University (UK) and includes 10 other partners: Technische Univ. Darmstadt (Germany), Univ. of Twente (The Netherlands), INRIA (project teams OBASCO , JACQUARD, TRISKELL and POP-ART), Vrije Univ. Brussels (Belgium), Trinity College Dublin (Irland), Univ. of Malaga (Spain), KU Leuven (Belgium), Technion (Israel), Siemens (Germany) and IBM UK.

With regard to technical integration work, the network was structured in four “laboratories:” a Languages Lab, a Formal Methods Lab, an Analysis and Design Lab, and an Applications Lab. OBASCO essentially took part in the first two labs whose main goals were a comprehensive meta-model and the corresponding implementation platform for the Languages Lab as well as a comprehensive semantic model and corresponding proof/analysis tools for the Formal Methods Lab. Furthermore, OBASCO coordinated the work of the four participating INRIA groups including Jacquard (Lille), Triskell (Rennes) and PopArt (Grenoble).

Overall funding of the network by the EU was 4.4 MEUR. OBASCO ’s share amounted to 200 KEUR. The web page is: http://www.aosd-europe.net.

In this final year, we have mainly worked on new formal properties of aspect-oriented systems, the implementation of dynamic aspect systems and an extension to the AWED system for explicitly distributed aspects.

The funding period of the network by the European Union has ended in Nov. 2008. We have participated in setting up an association that continues several key activities of the network (AOSD-Europe summer school, industry cooperations, mobility and dissemination activities). Most of the former network partners as well as two new partners have joined the association as of Nov. 2008.

8.3.2. STREP AMPLE

Participants: Jean-Claude Royer, Nicolas Anquetil, Hugo Arboleda, Pierre Cointe, Fabricio Fernandes, Jacques Noyé, Joost Noppen, Angel Nũnez, Mario Südholt.

The Aspect-Oriented, Model-Driven Product Line Engineering (AMPLE) project started on October 1, 2006 and will finish September 30, 2009. It involves the following partners: University of Lancaster (UK), Universidade Nova de Lisboa (Pt), École des Mines de Nantes (Fr), Darmstadt University (De), Universiteit Twente (Nl), Universidad de Málaga (Es), HOLOS (Po), SAP AG (De), Siemens (De). The aim of this project is to provide a Software Product Line (SPL) development methodology that offers improved modularisation of variations, their holistic treatment across the software lifecycle and maintenance of their traceability (forward and backward) during SPL evolution. 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 second year of the project, we have worked on the implementation of software product lines. Concretely, we have evaluated an extension of the aspect system CaesarJ by concurrent and distributed aspects in the language. The introduction of concurrent aspect, in the spirit of CEAOP has been prototyped and is currently experimented on the Smart Home system (Tente implementation). The integration of features for explicit distributed programming that is currently under design is inspired by our aspect system AWED.

In 2008, we have also participated in the design of a traceability software support and the second one devoted to its implementation. A prototype based on Eclipse is available and integrates various developments coming from at least 5 partners. The OBASCO project-team participated mainly in a trace view plugin (that provides graphical views, basic graph metrics and export of data) and the design decision rationale system. Several publications related to this tool have been presented at the ECMDA Traceability Workshop and Neptune conference:

- “A Model-Driven Traceability Framework to Software Product Line Development” [35] presents the traceability framework that was conceived and developed by the AMPLE members. In the paper we explain how the framework was designed, starting from a review of existing tools and their lacks, going on to the conception of the traceability metamodel, and terminating with the implementation of the tool.

- “Traceability for Model Driven, Software Product Line Engineering” [18] presents a traceability taxonomy created by the AMPLE team and different contributions related to it.
- Lastly [17] describes the general traceability metamodel elaborated in AMPLE and how to integrate in it design decision rationales.

The team organised in Nantes the event "Software Factories and Software Product Lines", a special issue of L’objet [40] was coordinated to collect some of the related contributions. The team members collaborates in a general introduction paper about the thematic [15] and a second paper dedicated to the AMPLE project [16].

We have also participated in work on two case studies. We have worked mainly on two implementations of a smart home application provided by Siemens AG, Germany. n this context, we have extended, in particular, the current MDE tool chain allowing fine-grained variability and fine-grained configuration [19].

Overall funding of the network by the EU is 3.78 million euros. OBASCO ’s share amounts to 370 KEUR. The web page is: http://www.ample-project.net for more details.

8.4. Collaboration with Foreign Research Groups

8.4.1. ECO-NET Project # 16293RG

Participants: Jean-Claude Royer, Jacques Noyé.

This project is funded by EGIDE and the partners are: the COLOSS team (LINA), the OBASCO project-team, the DSRG team (Charles University, Prague, Czech Republic) and the CSRL team (Universitatea Babes-Bolyai, Cluj-Napoca, Romania). The goal of the project is to establish a link from component code to component specifications. The focus is on extracting some abstract descriptions from Java programs, which paves the way for verification and compatibility checking. In 2008, two workshops have been organized: one in Nantes (May 2008) and a second one in Cluj-Napoca, Romania (September 2008). The partners have pursued the definition of a component metamodel and its implementation. The extraction of protocols and component architectures have been prototyped and preliminary experiments conducted.

8.4.2. CONICYT Chili - INRIA CORDIAL Project

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

CORDIAL stands for COncuRrency, Distribution, and Interactions in Aspect Languages. The objective of this two-year project CONICYT/INRIA, started in January 2008, is to advance the state of the art in concurrent and distributed aspect-oriented programming by leveraging the expertise of the two participating teams: OBASCO in Nantes and the newly-created PLEIAD laboratory, led by Éric Tanter, at *Universidad de Chile*. In accordance with the mobility plans, Johan Fabry, Éric Tanter, and Rodolfo Toledo, from PLEIAD, visited OBASCO in April, July, and October, respectively, while Angel Nuñez and Jacques Noyé visited PLEIAD in September and November. Some initial results of the project are reported in Sec. 3.7.

9. Dissemination

9.1. Animation of the Community

9.1.1. Animation

ADI: Mario Südholt has co-organized the international workshop “Aspects, Dependencies and Interactions” (ADI) [41] where work has been presented on a large range of semantic problems, in particular, stemming from dependencies among different aspects, as well as aspects and other application-structuring paradigms, such as feature-oriented programming.

DSAL: Jacques Noyé has co-organized the international workshop "Domain-Specific Aspect Languages" (DSAL 2008) [25].

SPL Day: Jean-Claude Royer and Nicolas Anquetil organised a meeting in Nantes June 16 with the title "Software Factories and Software product Lines", (see http://www.emn.fr/x-info/jsample/). Forty participants attended this event, mostly from industry. A special edition of the French journal "L'objet" has been prepared to publish selected articles presented at the event.

Embedded Software 2008: In the context of the Images et Réseaux cluster, OBASCO and LANDE co-organized a technical day in Rennes in December 2008 that was dedicated to generic and multi-concern solutions for embedded software development.

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.

9.1.2. Steering, journal, conference committees

P. Cointe: He is a member of the ECOOP and LMO steering committees (http://www.ecoop.org). He served in the CARI, DSL and NOTERE 2008 committees. Pierre Cointe was invited to the ERCIM Interlink Consolidation workshop in Cannes on 12-14 November 2008. He contributed to the Software intensive systems and new computing paradigms group.

T. Ledoux: was a program committee member of the international journal Annals of Telecoms (special issue "The Fractal Initiative"), CFSE-6 (6th French Conference on Operating Systems, March 2008, Fribourg, Switzerland), the IEEE International Workshop on Methodologies for Non-functional Properties in Services Computing (July 2008, Honolulu, Hawaii, USA), the 7th Workshop on Adaptive and Reflective Middleware (ARM'08, Leuven, Belgium, December 2008).

J.-M. Menaud: He is a member of the (RenPar/CFSE/Sympa) steering committees. He has served on the program committee of CFSE-6 (6th French Conference on Operating Systems, March 2008, Fribourg, Switzerland) and third Workshop on Virtualization in High-Performance Clusterand Grid Computing (VHPC'08) as part of Euro-Par 2008.

J. Noyé: He was a program committee member of LMO 2008 (*Langages et Modèles à Objets*), Montreal, March 2008, and SC 2008 (Software Composition), Budapest, March 2008. He is the co-editor of a special issue on Domain-Specific Aspect Languages to be published by the IET Software journal. He is a program committee member of LMO 2009, Nancy, March 2009.

J.-C. Royer: He is an editor-in-chief of the *RSTI L'Objet* journal, a member of the editorial board of the Journal of Object Technology (JOT), a member of the steering committee of RSTI (*Revue des Sciences et Technologies de l'Information*, Hermès-Lavoisier) and member of the program comittee of TSI. He was a member of the program comittee of CAL 2008 and CAL 2009.

M. Südholt: In 2008, he has been appointed to 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, since November 2008, co-located with the conference TOOLS Europe.

He has co-chaired the international workshop ADI'08 (Aspects, Dependencies and Interactions) at AOSD'08 and has served as the workshop chair of AOSD'08.

Finally, he has served on the program committees of the international conferences AOSD'08, ECOOP'08 and SC'08, as well as the international workshop on Runtime Verification'08. He acts as a workshop co-chair of ECOOP'09.

9.1.3. Thesis committees

P. Cointe: He was the reviewer of the PhD theses of M. Denker (University of Bern, 05/08) and N. Palix (Université de Bordeaux 1, 17/09/08).

J. Noyé and J.-C. Royer: were members of the PhD committee of Sebastian Pavel (Université de Nantes, 21/10/2008).

T. Ledoux: was a member of the PhD committees of Fabien Baligand (Ecole des Mines de Paris, 25/06/2008), Nagapraveen Jayaprakash (LSR-IMAG, 27/06/2008), and Franck Chauvel (UBS, 19/09/2008).

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 Engeenering* 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 *ARPEGE 2008* call.

J. Noyé: acted as an expert for the Israel Science Foundation.

9.2. Teaching

MSc EMOOSE. In September 1998, the team set up, in collaboration with a network of partners, an international Master of Science program EMOOSE (European Master of Science on Object-Oriented and Software Engineering Technologies). This program is dedicated to object-oriented software engineering in a broad sense, including component-based and aspect-oriented software development.

The program is managed by the team, with Jacques Noyé as the head, in cooperation with the *Vrije Universiteit Brussel* (VUB) and the courses take place in Nantes. The students receive a Master of Science degree of the *Vrije Universiteit Brussel* and a *Certificat d'études spécialisées de l'École des Mines de Nantes*. The tenth promotion graduated in August 2008 and the eleventh promotion started their first semester in October. 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 OBASCO members give lectures about new trends in the field of component and aspect-oriented software engineering.

AOSD-Europe Summer School. Pierre Cointe and Mario Südholt participated in this one week school in Darmstadt, Germany. They gave a lecture on *Reflection and AOP* and *Aspects for distributed and concurrent systems*, respectively.

Summer School on Programming Languages (PL 2008). Jacques Noyé participated in a two-day Summer School organized in Punta Arenas, Chile, by the PLEIAD laboratory of the University of Chile, together with the Chilean Computer Science Society (SCCC). He gave an introduction to *Process calculi* and talked about *Aspects, Processes, and Components*.

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 INRIA CR2 hiring committee, for the Rennes Bretagne Atlantique center.

T. Ledoux: He is a member of the board of the Regional Doctoral School STIM.

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), Distinguished Paper Award, ACM Press, March 2008, p. 60–71.

[5] 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 Professional, September 2004, p. 201-218.

[6] 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", vol. I, extended version of an article presented at AOSD'05, March 2006, p. 174–213.

[7] 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.

[8] 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)", to appear, March 2009.

[9] É. TANTER, J. NOYÉ, D. CAROMEL, P. COINTE. *Partial Behavioral Reflection: Spatial and Temporal Selection of Reification*, in "Proc. of the 18th ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA 2003), Anaheim, California, USA", R. CROCKER, GUY L. JR. STEELE (editors), vol. 38, n^{o} 11, ACM Press, October 2003, p. 27–46.

[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.

Year Publications

Doctoral Dissertations and Habilitation Theses

[11] F. BALIGAND. *Une Approche Déclarative pour la Gestion de la Qualité de Service dans les Compositions de Services*, Ph. D. Thesis, Ecole des Mines de Paris, Nantes, June 2008.

[12] S. PAVEL. *A Hierarchical Component Model with Interaction Protocols*, Ph. D. Thesis, Université de Nantes, Nantes, October 2008.

Articles in International Peer-Reviewed Journal

[13] 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", Special issue on Component-based architecture: the Fractal initiative, vol. 64, n^{o} 1-2, January-February 2009.

[14] 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.

Articles in National Peer-Reviewed Journal

[15] N. ANQUETIL, H. ARBOLEDA, F. FERNANDES, A. NUÑEZ, J.-C. ROYER. *Lignes de produits logiciels et usines logicielles*, in "RSTI - L'objet, Usines logicielles et lignes de produits logiciels", vol. 14, n^{o} 3, 2008, p. 15-31.

[16] N. ANQUETIL, J. NOPPEN, I. GALVÃO. *La traçabilité dans les lignes de produits logiciels*, in "RSTI - L'objet, Usines logicielles et lignes de produits logiciels", vol. 14, n^{o} 3, 2008, p. 47-57.

Articles in Non Peer-Reviewed Journal

[17] J. NOPPEN, J.-C. ROYER. *The Ample Project, Traceability of Software Product Line Development: Models and Uncertainty*, in "Revue du Génie Logiciel", n^{o} 85, June 2008, p. 43-48.

International Peer-Reviewed Conference/Proceedings

[18] N. ANQUETIL, B. GRAMMEL, I. GALVÃO, J. NOPPEN, S. S. KHAN, H. ARBOLEDA, A. RASHID, A. GARCIA. *Traceability for Model Driven, Software Product Line Engineering*, in "ECMDA Traceability Workshop (ECMDA-TW) 2008 Proceedings", ISBN 978-82-14-04396-9, June 2008, p. 77–86.

[19] H. ARBOLEDA, R. CASALLAS, J.-C. ROYER. *Using Transformation-Aspects in Model-Driven Software Product Lines*, in "Proc. of the 3rd International Workshop on Aspects, Dependencies, and Interactions (ADI'08) at ECOOP'08, Technical Report Katholieke Universiteit Leuven, No: CW 517, Paphos, Cyprus", July 2008, p. 46-56.

[20] A. ASSAF, J. NOYÉ. *Dynamic AspectJ*, in "Proceedings of the 2008 Dynamic Languages Symposium (DLS 2008), Paphos, Cyprus", J. BRICHAU (editor), ACM Press, July 2008.

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

[22] L. D. BENAVIDES NAVARRO, R. DOUENCE, F. HERMENIER, J.-M. MENAUD, M. SÜDHOLT. *Aspect-based patterns for grid programming*, in "Proc. of the 20th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD'08)", IEEE Press, October 2008.

[23] L. D. BENAVIDES NAVARRO, R. DOUENCE, A. NUÑEZ, M. SÜDHOLT. *LTS-based Semantics and Property Analysis of Distributed Aspects and Invasive Patterns*, in "Proc. of the 3rd International Workshop on Aspects, Dependencies, and Interactions (ADI'08) at ECOOP'08, Technical Report Katholieke Universiteit Leuven, No: CW 517, Paphos, Cyprus", July 2008, p. 36-45.

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

[25] T. CLEENEWERCK, J. NOYÉ, J. FABRY, A.-F. LEMEUR, É. TANTER. *Summary of the third workshop on Domain-Specific Aspect Languages*, in "DSAL '08: Proceedings of the 2008 AOSD workshop on Domain-specific aspect languages, Brussels, Belgium", T. CLEENEWERCK, J. NOYÉ, J. FABRY, A.-F. LEMEUR, É. TANTER (editors), ACM Press, 2008, p. 1–5.

[26] P.-C. DAVID, M. LÉGER, H. GRALL, T. LEDOUX, T. COUPAYE. *A Multi-Stage Approach for Reliable Dynamic Reconfigurations of Component-Based Systems*, in "Proceedings of the 8th IFIP International Conference on Distributed Applications and Interoperable Systems (DAIS'08), Oslo, Norway", LNCS, Springer Verlag, June 2008.

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

[28] S. DJOKO DJOKO, R. DOUENCE, P. FRADET. *Aspects Preserving Properties*, in "Proceedings of the ACM SIGPLAN Workshop on Partial Evaluation and Semantics-Based Program Manipulation (PEPM'08) co-located with POPL'08, San Francisco, CA, USA", ACM Press, January 2008, p. 135-145.

[29] S. DJOKO DJOKO, R. DOUENCE, P. FRADET. *Specialized Aspect Languages Preserving Classes of Properties*, in "Proc. of the Int. Conf. on Software Engineering and Formal Methods (SEFM'08), Cape Town, South Africa", IEEE Computer Society Press, November 2008.

[30] F. FERNANDES, R. PASSAMA, J.-C. ROYER. *Event Strictness for Components with Complex Bindings*, in "ISEC2009: Proceedings of the 2nd conference on India software engineering conference, Pune, India", To appear, ACM Press, February 2009, p. 5–14.

[31] F. FERNANDES, J.-C. ROYER. *The STSLib Project: Towards a Formal Component Model Based on STS*, in "Proc. of the 4th International Workshop on Formal Aspects of Component Software (FACS'07)", M. LUMPE, E. MADELAINE (editors), Electronic Notes in Theoretical Computer Science, n^{o} 215, June 2008, p. 131-149, http://dx.doi.org/10.1016/j.entcs.2008.06.025.

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

[33] J. NAGAPRAVEEN, T. COUPAYE, C. COLLET, P.-C. DAVID. *Flexible Reactive Capabilities in Component-Based Autonomic Systems*, in "EASE '08: Proceedings of the Fifth IEEE Workshop on Engineering of Autonomic and Autonomous Systems (ease 2008), Washington, DC, USA", IEEE Computer Society, 2008, p. 97–106.

[34] A. NUÑEZ, J. NOYÉ. *An event-based coordination model for context-aware applications*, in "10th International Conference on Coordination Models and Languages (COORDINATION 2008), Oslo, Norway", D. LEA, G. ZAVATTARO (editors), LNCS, vol. 5052, Springer-Verlag, June 2008, p. 232-248.

[35] A. SOUSA, U. KULESZA, A. RUMMLER, N. ANQUETIL, R. MITSCHKE, A. MOREIRA, V. AMARAL, J. ARAUJO. *A Model-Driven Traceability Framework to Software Product Line Development*, in "ECMDA Traceability Workshop (ECMDA-TW) 2008 Proceedings", ISBN 978-82-14-04396-9, 2008, p. 97–109.

[36] É. TANTER, J. FABRY, R. DOUENCE, J. NOYÉ, M. SÜDHOLT. *Expressive Scoping of Distributed Aspects*, in "Proceedings of the 8th ACM International Conference on Aspect-Oriented Software Development (AOSD'09), Charlottesville, Virginia, USA", A. MOREIRA, C. SCHWANNINGER (editors), to appear, ACM Press, March 2009.

National Peer-Reviewed Conference/Proceedings

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

[38] F. HERMENIER, X. LORCA, H. CAMBAZARD, J.-M. MENAUD, N. JUSSIEN. *Reconfiguration automatique du placement dans les grilles de calculs dirigée par des objectifs*, in "Proc. of 6ème Conférence Francophone sur les Systèmes d'Exploitation (CFSE06), Fribourg, Swiss", February 2008.

Books or Proceedings Editing

[39] T. CLEENEWERCK, J. NOYÉ, J. FABRY, A.-F. LEMEUR, É. TANTER (editors). *DSAL '08: Proceedings of the 2008 AOSD workshop on Domain-specific aspect languages*, ACM Press, Brussels, Belgium, 2008.

[40] J.-C. ROYER (editor). *RSTI - L'objet, Usines logicielles et lignes de produits logiciels*, vol. 14, n^{o} 3, Lavoisier, 2008.

[41] F. SANEN, R. CHITCHYAN, L. BERGMANS, J. FABRY, M. SÜDHOLT, K. MEHNER (editors). *Report on the Int. WS on Aspects, Dependencies and Interactions (ADI'07)*, LNCS 4906, Springer Verlag, February 2008, p. 75-90.

Research Reports

[42] K. GARCÈS, F. JOUAULT, P. COINTE, J. BÉZIVIN. *Adaptation of Models to Evolving Metamodels*, Research Report, n^{o} RR-6723, INRIA, November 2008.

[43] F. HERMENIER, X. LORCA, J.-M. MENAUD, G. MULLER, J. LAWALL. *Entropy: a Consolidation Manager for Clusters*, Research Report, n^{o} RR-6639, INRIA, September 2008, http://hal.inria.fr/inria-00320204/fr/.

Other Publications

[44] M. LACOUTURE. *A Generic Version of FScript: A Model-Driven Engineering Approach*, master EMOOSE, Masters thesis, Vrije Universiteit Brussel - Belgium Faculty of Sciences, Ecole des Mines de Nantes, August 2008.

[45] F. LEPAGE. *Environnement efficace pour automates symboliques : une approche par développement de prototype*, Mémoire d'Ingénieur CNAM, Masters thesis, Conservatoire National des Arts et Métiers des Pays de la Loire, December 2008.

References in notes

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

[47] 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.

[48] 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.

[49] 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.

[50] 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.

[51] A. BEUGNARD, J.-M. JÉZÉQUEL, N. PLOUZEAU, D. WATKINS. *Making Components Contract Aware*, in "Computer", vol. 32, n^{o} 7, July 1999, p. 38–44.

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

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

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

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

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

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

[58] G. Decker, O. Kopp, F. Leymann, M. Weske. *BPEL4Chor: Extending BPEL for Modeling Choreographies*, in "IEEE International Conference on Web Services (ICWS 2007)", IEEE Computer Society, 2007, p. 296–303, http://doi.ieeecomputersociety.org/10.1109/ICWS.2007.59.

[59] E. W. Dijkstra. *On the role of scientific thought*, in "Selected Writings on Computing: A Personal Perspective", Published in 1982, Springer Verlag, 1974, p. 60–66, http://www.cs.utexas.edu/users/EWD/transcriptions/EWD04xx/EWD447.html.

[60] R. Douence, P. Fradet, M. Südholt. *A framework for the detection and resolution of aspect interactions*, in "Proceedings of the ACM SIGPLAN/SIGSOFT Conference on Generative Programming and Component Engineering (GPCE'02)", LLNCS, preprint version is ftp://ftp.inria.fr/INRIA/publication/publi-pdf/RR/RR-4435.pdf, vol. 2487, Springer-Verlag, October 2002, p. 173–188.

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

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

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

[64] P. T. Eugster, P. A. Felber, R. Guerraoui, A.-M. Kermarrec. *The many faces of publish/subscribe*, in "ACM Computing Surveys", vol. 35, n^{o} 2, June 2003, p. 114–131.

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

[66] B. D. Fraine, M. Südholt, V. Jonckers. *A Formal Semantics of Flexible and Safe Pointcut/Advice Bindings*, Technical report, n^{o} SSEL 02/2007/a, Vrije Universiteit Brussel, October 2007.

[67] 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.

[68] E. Gamma, R. Helm, R. Johnson, J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison Wesley, Massachusetts, 1994.

[69] G. Kiczales, E. Hilsdale, J. Hugunin, M. Kersten, J. Palm, W. G. Griswold. *An Overview of AspectJ*, in "ECOOP 2001 — Object-Oriented Programming 15th European Conference, Budapest Hungary, Berlin", J. L. Knudsen (editor), Lecture Notes in Computer Science, AspectJ web site: http://aspectj.org, vol. 2072, Springer-Verlag, June 2001, p. 327–353.

[70] G. KICZALES. *Aspect Oriented Programming*, in "Proc. of the Int. Workshop on Composability Issues in Object-Orientation (CIOO'96) at ECOOP", Selected paper published by dpunkt press, Heidelberg, Germany, July 1996, http://trese.cs.utwente.nl/cioo96.

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

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

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

[74] X. LEROY. *Manifest types, modules, and separate compilation*, in "Proceedings of the 21st ACM SIGPLAN-SIGACT Symposium on Principles Of Programming Languages, Portland, Oregon, USA", ACM Press, January 1994, p. 109-121.

[75] M. MCILROY. *Mass produced software components*, in "Proceedings of the NATO Conference on Software Engineering, Garmish, Germany", P. NAUR, B. RANDELL (editors), NATO Science Committee, October 1968, p. 138-155, http://www.cs.dartmouth.edu/~doug/components.txt.

[76] 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.

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

[78] M. MERNIK, J. HEERING, A. M. SLOANE. *When and How to Develop Domain-Specific Languages*, in "ACM Computing Surveys", vol. 37, n^{o} 4, December 2005, p. 316-344, http://doi.acm.org/10.1145/1118890.1118892.

[79] L. MIKHAJLOV, E. SEKERINSKI. *A study of the fragile base class*, in "ECOOP'98 - Object-Oriented Programming - 12th European Conference, Brussels, Belgium", E. JUL (editor), Lecture Notes in Computer Science, vol. 1445, July 1998, p. 355-382.

[80] 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.

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

[82] 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.

[83] 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.

[84] 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.

[85] 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.

[86] F. PLASIL, S. VISNOVSKY. *Behavior Protocols for Software Components*, in "Transactions on Software Engineering", vol. 28, n^{o} 9, January 2002, http://nenya.ms.mff.cuni.cz/publications/tse2002.pdf.

[87] 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.

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

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

[90] 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.

[91] 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.

[92] 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.

[93] 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.

[94] 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.

[95] 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.

[96] 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.