INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team Jacquard

Weaving of Software Components

Futurs

THEME COM

Table of contents

1. Team **1**
2. Overall Objectives **2**
2.1. Overall Objectives 2
2.1.1. J.M. Jacquard and Weaving Machines 2
2.1.2. Context and Goals 2
3. Scientific Foundations **2**
3.1. Weaving of Software Components 2
3.2. OpenCCM 3
3.2.1. Open Middleware for the CCM 3
3.2.2. Open Containers 4
3.2.3. Open Environment 4
3.3. Aspect-Oriented Design of Dynamic Component Assemblies 5
3.3.1. Early Aspects 5
3.3.2. Aspects at Design-time 5
3.3.3. Aspects at Run-time 5
4. Application Domains **6**
4.1. Application Domains 6
5. Software **6**
5.1. AOKell 6
5.2. FAC 6
5.3. Fraclet 7
5.4. Fractal Deployment Framework 7
5.5. Fractal Explorer 8
5.6. GoTM 8
5.7. Java Aspect Components 9
5.8. OpenCCM 9
5.9. SafArchie Studio 10
5.10. Spoon 10
6. New Results **11**
6.1. Open Middleware for the CCM 11
6.1.1. Generic Framework for Large Scale Distributed Deployment 11
6.1.2. Distributed Autonomous Component-based Architectures 12
6.1.3. Component-Based Software Framework for Building Transaction Services 13
6.1.3.1. Overview of the GoTM Activity 13
6.1.3.2. Building Heterogeneous Transaction Services 13
6.1.3.3. Supporting Dynamic Adaptation of 2-Phase Commit Protocols 14
6.1.4. Leveraging Component-Oriented Programming Using Attribute-Oriented Programming 14
6.1.4.1. Leveraging Fractal-Based Developments 14
6.1.4.2. Supporting Various Component Models 15
6.2. Aspect-Oriented Design of Dynamic Components Assemblies 15
6.2.1. Early Aspects and Model Driven Engineering 15
6.2.2. Aspects at Design Time 15
6.2.3. Logic Pointcut Languages for Object-Oriented Programs 16
6.2.4. Aspects and Components for Software Architectures 17
6.2.5. Complex Aspects 18
6.2.6. Transformation Languages and Tools 18
7. Contracts and Grants with Industry **19**
7.1. France Telecom 19
7.2. NorSys 19

8. Other Grants and Activities **19**
8.1. Regional Initiatives 19
8.1.1. MOSAIQUES 19
8.2. National Initiatives 20
8.2.1. ARA REVE 20
8.2.2. ARC COA 20
8.2.3. RNTL FAROS 20
8.2.4. RNTL JOnES 20
8.3. European Initiatives 21
8.3.1. AOSD-Europe 21
8.3.2. CALA 21
8.3.3. ITEA S4ALL 21
8.4. International Initiative 21
8.4.1. LAFMI AProSec 21
8.4.2. ICT-Asia 21
8.4.3. ObjectWeb 22
8.4.4. OMG 22
9. Dissemination **22**
9.1. Scientific Community Animation 22
9.1.1. Examination Committees 22
9.1.2. Journals, Conferences, Workshop 23
9.2. Teaching 26
9.3. Administrative Responsibilities 26
10. Bibliography **27**

1. Team

Jacquard is a joint project between INRIA, CNRS and Université des Sciences et Technologies de Lille (USTL), via the Computer Science Laboratory of Lille : LIFL (UMR 8022).

Team Leader

Jean-Marc Geib [Professor, USTL, HdR]

Administrative Assistant

Axelle Magnier [Project Assistant INRIA, until August 31th 2006]
Corinne Davoust [Project Assistant INRIA, since September 1st 2006]

Staff member INRIA

Laurence Duchien [Research Associate (secondment INRIA), since September 1st 2006, HdR]
Philippe Merle [Research Associate (CR1), INRIA]
Renaud Pawlak [Research Associate (CR2), INRIA, until September 30th 2006]
Lionel Seinturier [Research Associate (secondment INRIA), until August 31th 2006]

Staff member LIFL

Laurence Duchien [Professor USTL, until August 31th 2006, HdR]
Anne-Françoise Le Meur [Associate Professor, USTL]
Lionel Seinturier [Professor, USTL, since September 1st 2006, HdR]

Ph. D. Student

Dolorès Diaz [NorSys CIFRE grant]
Jérémy Dubus [MESR grant]
Guillaume Dufrêne [RNTL FAROS, since October 1st 2006]
Frédéric Loiret [CEA grant]
Naouel Moha [University of Montreal]
Carlos Francisco Noguera Garcia [IST AOSD Network of Excellence grant]
Nicolas Pessemier [France Télécom grant]
Ales Plsek [INRIA CORDI grant, since October 1st 2006]
Romain Rouvoy [INRIA-Région grant, until September 30th 2006]
Guillaume Waignier [MESR grant, since October 1st 2006]

Post-doctoral Fellow

Johan Brichau [Post-doctoral fellow - CNRS grant, until August 31th 2006]
Maja D'Hondt [Post-doctoral fellow - ERCIM grant, until June 30th 2006]
Johan Fabry [Post-doctoral fellow - INRIA grant, since September 1st 2006]

Invited Scientist

Denis Conan [INT, January-June 2006]
Théo D'Hondt [Vrije Universiteit Brussel (VUB), one week in March 2006]
Vincent Englebert [University of Namur, one week in March 2006]
Roberto Gomez [Instituto Tecnológico y de Estudios Superiores de Monterrey (Mexico), one week in December 2006]
Kim Mems [Catholic University of Louvain (UCL), one week in May 2006]
Tom Mems [University of Mons, December 2006 - October 2007]

Project technical staff

Nicolas Dolet [Project staff - ITEA S4ALL and RNTL JOnES, since March 1st 2006]
Areski Flissi [Technical staff CNRS]
James Manley [Project staff - ITEA S4ALL, since March 1st 2006]
Nicolas Petitprez [Project staff INRIA - until October 31th 2006]
Missi Tran [Project staff INRIA - July -December 2006]

2. Overall Objectives

2.1. Overall Objectives

Keywords: *Aspect-Oriented Programming (AOP)*, *Component Weaving*, *Component Models*, *Component-Based Adaptive Middleware (CBAM)*, *Integrated Tools for Production and Exploitation of Software Components*, *Model-Driven Software Engineering (MDSE)*, *Run-time Containers*, *Separation of Concerns (SoC)*, *Software Architecture*.

2.1.1. J.M. Jacquard and Weaving Machines

One of the first historical steps towards programming appeared in 1725 on a weaving machine. The French "Lyonnais" Basile Bouchon first gives instructions to a weaving machine using a perforated paper. His assistant Mr Falcon replaces the fragile paper by more robust perforated cards. After that, Mr Vancanson replaces the cards by a metallic cylinder and a complex hydraulic system, which gives the machine a cyclic flow of instructions, a program!

But History keeps in mind Joseph-Marie Jacquard who creates and commercialises the first automatic weaving machine during the beginning of 19th century. The precision of the machine allows Joseph-Marie Jacquard to design a program that weaves his own face on a fabric. Joseph-Marie Jacquard's innovations have greatly contributed to the first steps of computer science with the perforated cards to support programs. The idea of independent programs for a programmatic machine was born!

2.1.2. Context and Goals

The Jacquard project focuses on the problem of designing complex distributed applications, i.e. those composed of numerous cooperative and distributed software components, which are constrained by various requirements, such as persistency, security and fault tolerance. We want to investigate the ability of software engineers to produce new component-oriented platforms and new methodological and technical approaches to design and exploit these applications. In particular, we explore the use of component models, separation of concerns and weaving in the different phases of an application's life cycle (i.e. modelling, design, assembling, deployment, and execution). Our goal is to produce fully functional platforms and tools. Finally, we are members of a number of standardization organizations (OMG) and the open source software world (ObjectWeb).

3. Scientific Foundations

3.1. Weaving of Software Components

One of the challenges for software components is to build new models and platforms to allow large scale interoperability of components for designing complex distributed applications. Actually, some models already exist: Enterprise Java Beans by Sun Microsystems, .NET by Microsoft and the CORBA Component Model in the CORBA3 OMG standard [93]. These models and platforms are clearly not satisfactory because of the lack of functional completeness and interoperability. Moreover, these industrial propositions only deal with a lot of technical problems to capture the component software notion, but mainly forget the need to manipulate the models of components and applications independently of the technical aspects. This point has been recently tackled by OMG with its Model Driven Architecture (MDA) initiative [92], [98]. We agree that these points (Component Models, Component oriented Platforms and Model Driven Engineering) lead to new research problems with the goal to produce a better integrated product line, from analysis to exploitation, for component-based applications.

Jacquard members have a large amount of research experience in two computer science domains related to the goal of the project: Jean-Marc Geib and Philippe Merle have some important contributions in the Distributed Object based Platforms area [70], Laurence Duchien and Lionel Seinturier on specifications and use of separation of concerns for complex applications. For example, we can quote the contributions to the OMG standardization work with the CorbaScript language [89] (proposed to the Scripting Language for CORBA RFP, and accepted as the CORBA Scripting Language chapter of CORBA3 [80]) and with the CCM (CORBA Component Model) chapter for which we lead the response group and the revision task force. Other examples are the JAC (Java Aspect Component) platform, one of the leading platforms for dynamic weaving of aspects [94], [96], [97].

We aim to join these experiences to design and produce an ambitious new platform for component-based complex applications with new methodological and technical traits for structuring the large set of strongly related problems in supporting these applications. Models, platforms and applications have to benefit from new open middleware using separation of concerns and weaving. Our contributions want to understand how a better structure of models and platforms can result in better software for complex applications.

For the next four years, the project's goals are:

- to produce a full platform for the CCM model. This platform, called OpenCCM, has to contribute to the OMG standardization work. Moreover it will provide new adaptable containers allowing the weaving of system aspects, dynamically following the application requirements. It will also provide an integrated environment to manipulate, deploy and exploit assemblies of components.
- to define a complete design and technical environment for assembling of components and aspects, via a dedicated modelling tool for composition and a dynamic aspects oriented platform, which will be a next step of our JAC platform.

3.2. OpenCCM

This part of the project deals with the design and the production of new tools for component based platforms. This work was initiated in the Computer Science Laboratory of Lille (LIFL) and is now one of the projects of the ObjectWeb Consortium [73] under the name OpenCCM. Our goal is a complete platform for the OMG's CORBA Component Model (CCM). We want to fully capture all the aspects of this standard and contribute to it. Our ambition is to produce the first referenced CCM platform in an open source format. Actually OpenCCM is already a LGPL software accessible at http://openccm.objectweb.org. Beyond this production we aim to investigate three points as research topics: open the platform to allow extensibility and adaptability, open the run-time containers to weave non-functional aspects, and give the capability to freely assemble components in an open environment. These three points are detailed in the next sections. This work is related to other works on open middleware: the Fractal model [67] for component middleware (ObjectWeb, INRIA Sardes project, France Telecom), reflexive middleware approaches (Dynamic TAO [78], Flexinet [72], OpenCorba [82], OpenORB [99]), adaptable middleware approaches (ARCAD RNTL project [83]), virtual machines (VVM) and QoS driven Middleware [74].

3.2.1. Open Middleware for the CCM

The OpenCCM project proposes an open framework to produce and exploit CORBA Components. One can specify such a component in the new OMG IDL3 language, which is an extension of the old CORBA IDL2 language. The framework can produce IDL2 schemas from IDL3 descriptions, and the associated stubs for various programming languages (Java, C++, IDLscript, etc.) [87]. The framework is itself composed of reusable components around an IDL3 global repository. This architecture is open and extensible. The components are written in the Java language and are also CORBA components, so that they can be assembled to create several configurations. Therefore the platform can be instantiated in several ways for middleware such as ORBacus, OpenORB or Borland Enterprise Server.

Current work plans to complete the framework with the Component Implementation Definition Language, the Persistent State Definition Language, and the JORM framework. This will allow the platform to automatically generate containers with persistency capabilities. We work also on the assembly and packaging tools using the XML descriptors of CCM, and we also work on the transformation tools towards C++.

3.2.2. Open Containers

A major goal of component based platforms is to be able to separate functional aspects (ideally programmed by an expert of the domain addressed) from the non-functional aspects (ideally programmed by an expert of computer system techniques). This separation can be implemented by a technical separation between the components (functional aspects) and the containers (non-functional aspects). A container hosts components, so that the components inherit the non-functional aspects of the container.

Actually containers (such as the EJB or CCM containers) can only contain a limited set of non-functional aspects (activation/termination, communications and events, security, transactions and persistency). These containers are not extensible, neither statically nor dynamically. So they cannot respond to specific needs such as fault tolerance, replication, load balancing, real-time or monitoring.

In contrast to this, we plan to design open containers. We investigate a generic model for containers and then weaving mechanisms which will allow an application to specify particular needs. An application will be able to reclaim the deployment of well-fitted containers. We work on a specific API to develop non-functional aspects for our containers. In a first step we have to specify a large number of non-functional aspects to find the way to compose them. Non-functional aspects can be seen as interceptors, so we work on composition of interceptors to produce containers. In a second step we investigate the possibility to dynamically manipulate the containers to change the configuration of non-functional aspects.

3.2.3. Open Environment

An open environment for component-based applications has to deal with several problems. For instance, we have to allow assemblies and deployment on demand. In this part we plan three goals: a virtual machine for programming distributed deployments, a trader of components to realize assemblies from 'off the shelf components, a repository to manipulate and drive assemblies of components.

Current middleware proposes fixed deployment strategies which are not adaptable to specific needs. These deployment tools are mainly 'black boxes' and ad-hoc for a particular environment. In the CCM context we can exploit the XML based OSD language which is used to describe assemblies. This is a good basis to describe deployments. But the CCM does not define an API to control the deployment, and today the associated tools have not be realized in an open manner. Actually we work on a set of operations to deploy OSD assemblies. We investigate several useful properties (such as optimised deployment, parallel deployment, fault tolerant deployment, transactional deployment) implemented by these operations. This will lead to an open API for adaptable deployment strategies [85], [86]. We plan to use IDLscript to specify such strategies.

Assemblies can be constructed on demand with 'Components Off The Shelves'. We work on this point with our TORBA environment [81]. Within TORBA we can instantiate components for trading from trading contracts (specified in our TDL - Trading Description Language). This is the basis for an open infrastructure for components brokering that we plan to investigate here.

In an open framework for components we have to manipulate assemblies in all the phases of the design work and also during execution. Assemblies have to be manipulated by various users, each with his/her own concerns (e.g., assemble, deploy, distribute, non-functional aspects set-up, monitoring). We plan to construct a global repository for all these activities. Moreover this repository has to be opened for new activities. In this way we want to define an environment which allows for the definition, at a meta level, of the different concerns that we want to exist on the repository [84]. Then the environment will be able to automatically generate a new view on the repository to capture the specified activity [79].

3.3. Aspect-Oriented Design of Dynamic Component Assemblies

The behaviour of a complex application in an open environment is difficult to specify and to implement because it has to evolve in accordance with the context. Changes can occur in an asynchronous manner and the behaviour has to be adapted without human actions and without stopping the application. A language to specify an assembly of components has to capture these dynamic aspects. A platform which supports the assembly at run-time also has to be able to respond to the needed changes. In this part of the project we plan to investigate three directions.

The first one deals with the study of separation of concerns from the first steps of analysis to the implementation. The goal is to trace the evolution of different concerns in these various stages. The second goal is related to the dynamic features of the Architecture Description Languages (ADL) [75], [88]. The last goal focuses on Aspect-Oriented Programming [69] [76] [95] in which one can capture a specific concern of an application.

Finally, this project part enhances specifications of component assemblies with the goal of designing adaptable applications. We introduce integration contracts for specifying the impact of components on the application and its context [62], [66]. Our approach is based on AOP to specify connection and integration schemas.

3.3.1. Early Aspects

Business applications are faced with two main challenges. On one hand, they are mostly developed using an iterative process where business functionalities are added to the core application as the project requirements evolve [55]. On the other hand, the non-functional requirements (in terms of security, remote communication and transaction, data persistence, etc.) are also high and need to be incorporated as seamlessly as possible. Both the component-based and the aspect-oriented approaches separatively provide directions for these challenges. However there is no integrated software processes that takes both into account. The goal of this work is thus to propose such a process and some tools to support it. This process starts at the early stages of analysis and the tools provide features to trace the evolution from user requirements until deployment and run-time. This work is done in the context of Dolores Diaz's PhD thesis [68].

3.3.2. Aspects at Design-time

Software architects and designers use reasoning frameworks to iteratively integrate functional and non-functional concerns into their projects, and to adapt them to unforseen functional or non-functional requirements. To assist them, analysis methods support modelling and verification tools from functional to technical architecture. Their main advantages are to model large-scaled distributed systems that require interoperability between system parts and the separation of concerns between business functionality and communication mechanisms.

However, no standard and universal definition of software architecture has been accepted by all the community [56], [57], [64], [88]. Various points of view on different studies yield several approaches. These approaches focus on only one or two concerns such as component interface specification, behavioural analysis or software reconfiguration. So we argue that, in order to increase the benefits of software architecture approaches, one may need to use an architecture centric approach with a global reasoning: from software architecture design to software architecture management to software architecture building, deployment and refinement. However, these different concerns of a software architecture definition must be kept consistent.

Our first goal is to propose enhancements of a component model for specifying dynamic evolution of a software architecture. It concerns three points of view: structural, functional and behavioural points of view [91], [61], [59]. We use the Model Driven Architecture approach with Context Independent Model and Context Specific Model. Our second goal is to introduce non-functional aspects and connections between components and containers in languages for software architectures. We extend contracts between components to contracts between components and non-functional components [60].

3.3.3. Aspects at Run-time

In distributed environments, applications run in an open context. They use networks and their associated services for which quality of service is not always guaranteed and may change quickly. In these environments,

several concerns must be considered, including fault tolerance, data consistency, remote version update, runtime maintenance, dynamic lookup, scalability, lack of rate. Addressing these issues may require dynamic and fast reconfiguration of distributed applications [90], [100], [101].

We have defined the Java Aspect Components (JAC) framework for building aspect-oriented distributed applications in Java [94], [96], [97]. Unlike other languages such as AspectJ, JAC allows dynamic weaving of aspects (aspects can be woven or unwoven at run-time) and proposes a modular solution to specify the composition of aspects. We defined an aspect-oriented programming model and the architectural details of the framework implementation. The framework enables extension of application semantics for handling well-separated concerns. This is achieved with a software entity called an aspect component (AC). ACs provide distributed pointcuts, dynamic wrappers and metamodel annotations. Distributed pointcuts are a key feature of our framework. They enable the definition of crosscutting structures that do not need to be located on a single host. ACs are dynamic: They can be added, removed, and controlled at runtime. This enables our framework to be used in highly dynamic environments where software adaptation is needed.

4. Application Domains

4.1. Application Domains

The Jacquard project addresses the large problem of designing complex distributed applications composed by numerous cooperative and distributed software components. Our application domains are numerous. First, our component models and platforms target information systems. These software need functional and non functional properties and they must evolve. Second, component models tackle several specific domains needing adaptability of process context such as mobility or ubiquitous computing. We apply our work in transportation or communication domains, for example in the MOSAIQUES project or the AOSD NoE. Finally, we participate in platform definitions for grid computing.

5. Software

5.1. AOKell

Keywords: *AOP*, *Fractal*.

Participants: Nicolas Pessemier, Lionel Seinturier [correspondant].

AOKell is a reflective and open implementation of the Fractal component model. So far, software components are used in various application domains (for embedded systems to e-commerce web sites). However the model of a component is different in each of these domains with solutions such as EJB, CCM, .NET, Accord/UML, ArchJava, OpenCOM, K-Component, etc. This variety hinders the broad adoption of component based techniques. The challenge we are addressing with AOKell is to design and implement a component model adaptable to various application domains. To achieve this goal, AOKell is a reflective component model with two dimensions: the business dimension and the control dimension. The former is concerned with the programming of application functionalities, while the latter is concerned with control and non-functional services needed by the application. The control dimension is fully programmable in AOKell with the same artefacts (component, binding, components assembly) that are available for the business dimension. The integration of these two dimensions is achieved with aspect-oriented programming. The work on AOKell is conducted in the context of a France Telecom R&D research grant.

AOKell is LGPL open source software available at http://fractal.objectweb.org/aokell

5.2. FAC

Keywords: *AOP*, *Fractal*.

Participants: Nicolas Pessemier [correspondant], Lionel Seinturier.

FAC (Fractal Aspect Component) is an extension of the Fractal component model for aspect-oriented programming. The purpose of FAC is to provide a programming model where components and aspects are first-class entities which can be assembled and composed seamlessly by designers and developers. FAC is innovating in the sense that, on the international stage, aspects have been studied so far only at the object level. We believe that the issues of code tangling and scattering exist at a higher level of granularity that of components. With FAC, we wish to provide one of the first models and its corresponding implementation for modularizing crosscutting concerns in component-based applications. The longer term goal of FAC is also to provide a unified framework where crosscutting concerns can be modularized at different levels of granularity: object, component and architecture. The work on FAC is conducted in the context of a France Telecom R&D research grant and is the subject of Nicolas Pessemier's PhD thesis.

FAC is LGPL open source software available at http://www.lifl.fr/ pessemie/FAC/

5.3. Fraclet

Keywords: *Attribute-Oriented Programming*, *Component-Based Programming*, *Fractal*.

Participants: Philippe Merle, Renaud Pawlak, Nicolas Pessemier, Romain Rouvoy [correspondant].

Fraclet is an attribute-oriented programming model for developing reliable Fractal components.

Fraclet is both an implementation of the Fractal specifications conformed to the level 0.1, and an annotation framework for the Fractal component model. Fraclet is composed of eight annotations and five plugins to generate automatically various artifacts required by the Fractal component model. Annotations provide an abstract way to describe the component meta-information directly in the source code of the content class. Fraclet plugins generate either Fractal component glue, Fractal ADL definitions or Monolog configurations.

Two implementations of the Fraclet annotation framework exist: Fraclet-XDoc and Fraclet-Annotation.

- Fraclet-XDoc uses the XDoclet generation engine and Javadoc-like annotations to produce the various artifacts required by the Fractal component model.
- Fraclet-Annotation uses the Spoon transformation tool and Java 5 annotations to enhance the handwritten program code with the non-functional properties of the Fractal component model.

Fraclet is used into several software projects based on the Fractal component model, i.e. the Jacquard team GoTM and Fractal Deployment Framework projects, the Denis Conan's COSMOS project, and the ObjectWeb PEtALS project (see http://petals.objectweb.org).

Fraclet is LGPL open source software available at http://fractal.objectweb.org/fraclet.

5.4. Fractal Deployment Framework

Keywords: *Component*, *Deployment*, *Middleware*.

Participants: Nicolas Dolet, Areski Flissi, James Manley, Philippe Merle [correspondant].

Fractal Deployment Framework (FDF) is a component-based software framework to facilitate the deployment of distributed applications on networked systems. FDF supports any kind of deployment activities like uploading, installing, configuring, starting, stopping and uninstalling any software. For that, FDF is composed of a high level deployment language, a library of deployment components, and a set of end-user tools.

The FDF language is a kind of high level scripting language allowing end-users to describe their deployments (i.e., hosts of the target system and software to deploy on them). This language relies on a library of deployment components wrapping various file transfer protocols (e.g., FTP, SCP), remote access protocols (e.g., TELNET, SSH), Unix and Windows shells, Internet-related notions (e.g., hostname, port, host), and software (e.g., middleware, services, daemons, application servers, application components). FDF also provides a graphical user interface allowing end-users to load their deployment descriptions, execute and manage them.

Currently, FDF supports the deployment of Java Runtime Environments (JRE), Java 2 Enterprise Edition (J2EE) application servers as JOnAS and JBoss, Java Business Integration (JBI) containers as PEtALS, OSGi gateways like OSCAR, and CORBA component-based application servers like OpenCCM. FDF has been successfully experimented for the deployment of OpenCCM middleware and CORBA component-based applications on one thousand nodes of Grid5000, the french grid infrastructure dedicated to computer science research.

FDF is designed with the ObjectWeb Fractal component model, and is implemented on top of its Java-based tools (Julia or AOKell, Fraclet, Fractal Explorer).

FDF is LGPL open source software available at http://gforge.inria.fr/projects/fdf.

5.5. Fractal Explorer

Keywords: *Fractal Component-Based Software Framework*, *Graphical User Interface*, *Management Console*.

Participant: Philippe Merle [correspondant].

Fractal Explorer is a generic Fractal component-based software framework to build Java-based graphical explorer and management consoles.

Fractal Explorer is composed of the Explorer Description Language, the plug-in programming interface, and the Fractal component-based explorer framework. The Explorer Description Language (a XML DTD) allows users to describe at a high level the configuration of graphical explorer consoles to build (i.e. icons, menu items and panels associated to resources to explore/manage) according to end-user roles. Reactions associated to these described graphical elements can be implemented by Java classes or BeanShell scripts which must be conform to the plug-in programming interface. Finally, the explorer framework implements the interpretation of explorer configurations and executes plug-in classes/scripts according to users' interactions. This framework is implemented as an extensible set of software components conform to the ObjectWeb Fractal component model defined by INRIA and France Telecom. Moreover a set of plug-ins is already provided to explore and manage any Java object or Fractal component.

Fractal Explorer is already reused and customized by our Apollon, FAC, GoTM, and OpenCCM software to provide respectively explorer consoles for XML documents, Fractal aspect components, component-based transaction services and CORBA objects/components.

Fractal Explorer is LGPL open source software available at http://fractal.objectweb.org/fractalexplorer.

5.6. GoTM

Keywords: *Component-Based Software Framework*, *Middleware Transaction Services*.

Participants: Philippe Merle, Romain Rouvoy [correspondant].

GoTM is a Fractal component-based software framework to build middleware transaction services.

GoTM is composed of an extensible set of Fractal components providing basic building blocks (Transaction, Resource, Coordination, Concurrency, etc.) to build various transaction models and services (OMG OTS, Sun JTA, etc.) [13]. This framework provides personalities to build Java Transaction Services (JTS) and Object Transaction Services (OTS). To deal with the transaction demarcation concern, GoTM provides the Open Transaction Demarcation Framework (OTDF) as a dedicated personality. The JTS and OTDF personalities are currently integrated into the ObjectWeb JOnAS application server.

The GoTM framework can be used to build heterogeneous transaction services [13]. Heterogeneous services support multiple transaction standards simultaneously without the performance degradation implied by the use of a coordination protocol.

The GoTM framework implements various 2-Phase Commit protocols (e.g., 2PC, 2PCPA, 2PCPC, etc.). It dynamically adapts the active transaction protocol to provide better transaction completion time depending on the commit/abort rate statistics [42].

The GoTM component-based software framework is designed on top of the ObjectWeb Fractal component model and is implemented on top of the ObjectWeb Julia reference implementation. Since recently, GoTM uses the AOKell software supported by the Jacquard project. AOKell implements the Fractal component model using Aspect Oriented Programming (AOP). This choice provides to GoTM the complementarity of technologies like aspects and components to build middleware solutions with added values. GoTM also uses the Fractal Explorer software to build its management tool. GoTM provides various explorer plugins (GoTM, JTS, OTS, etc.). Thus, GoTM consoles can manage heterogeneous GoTM transaction services. Finally, the GoTM OTS personality aims to be integrated in the OpenCCM software developed in the Jacquard project.

On the topic of Transactions and Components, the GoTM and AOKell projects are also collaborating to provide transparent transaction support to Fractal components.

GoTM results from the activities done in the Transaction Working Group created in the context of the ITEA OSMOSE project. GoTM will be integrated in the next Enterprise Service Bus (ESB) solution developed by the ObjectWeb consortium in the context of the RNTL JOnES project.

GoTM is LGPL open source software available at http://gotm.objectweb.org.

5.7. Java Aspect Components

Keywords: *Dynamic Weaving*, *Java Aspect Components*.

Participants: Renaud Pawlak [correspondant], Lionel Seinturier.

JAC (Java Aspect Components) is a project consisting out of developing an aspect-oriented middleware layer. The current version of JAC is 0.12.1. Current application servers do not always provide satisfying means to separate technical concerns from the application code. Since JAC uses aspect-orientation, the complex components are replaced by POJOs (Plain Old Java Objects) and non-functional concern implementations that are usually wired deep into the container implementations are replaced by loosely-coupled, dynamically pluggable aspect components. JAC aspect components provide: seamless persistence (CMP) that fully handles collections and references, flexible clustering features (customisable broadcast, load-balancing, data-consistency, caching), instantaneously defined users, profile management, access rights checking, and authentication features.

JAC is LGPL open source software available at http://jac.objectweb.org

5.8. OpenCCM

Keywords: *CORBA Component Model*, *Component-Based Middleware*.

Participants: Areski Flissi, Philippe Merle [correspondant].

OpenCCM is a middleware platform for distributed applications based on CORBA components.

OpenCCM stands for the Open CORBA Component Model Platform: The first public available and open source implementation of the CORBA Component Model (CCM) specification defined by the Object Management Group (OMG). The CORBA Component Model (CCM) is the first vendor-neutral open standard for Distributed Component Computing seamlessly supporting various programming languages, operating systems, networks, CORBA products and vendors. The CCM is an OMG's specification for creating distributed, server-side scalable, component-based, language-neutral, transactional, multi-user and secure applications. Moreover, one CCM application can be deployed and run on several distributed nodes simultaneously.

OpenCCM allows users to design, implement, compile, package, assemble, deploy, install, instantiate, configure, execute, and manage distributed CORBA component-based applications. For these purposes, OpenCCM is composed of a set of tools, i.e., UML and OMG IDL model repositories, compilers, code generators, a graphical packaging and assembling tool, a distributed deployment infrastructure, extensible containers integrating various services (communication, monitoring, transaction, persistency, security, etc.), and a graphical management console.

OpenCCM is LGPL open source software available at http://openccm.objectweb.org.

5.9. SafArchie Studio

Keywords: *Transformation process*, *architecture-centric IDE*.

Participant: Missi Tran-Anh [correspondant].

SafArchie Studio is a software architecture-centric IDE that implements the SafArchie (Safe Architecture) model. It is composed of three main parts. The first one is a set of extensions for ArgoUML to transform the UML case tool into a software architecture case tool. It allows the architect to graphically design a component based software architecture. The second part analyses software architecture consistency. This consists of checking the structural compatibility between bound ports and the behavioral compatibility between bound components with respect to the SafArchie model. The third part is a skeleton generator for the ArchJava language or the Fractal component model (more precisely France Telecom's official implementation: Julia). This generator provides a first level connection between the design and the implementation of a component-based system. The longer term goal of SafArchie Studio is to provide a unified IDE to design, transform and control the evolution of component-based applications. The work on SafArchie Studio was a part of Olivier Barais's PhD thesis [58] and is mainly developed by Missi Tran-Anh and Olivier Barais.

TranSAT tools are designed as extensions of SafArchie Studio (see above). They aim at providing IDE support to allow the architect to stepwise design a software architecture. TranSAT is associated with three main tools:

- new diagram editors to specify new software architecture patterns
- a compiler for the software architecture pattern. It performs static analyses to guarantee that the pattern will not compromise the consistency of the software architectures that will be modified.
- a transformation processor. It integrates a new pattern into a software architecture.

The transformation processor includes a first module to find all the join points that satisfy the join point mask constraints, a second module to dynamically verify the consistency of the resulting software architecture and finally a transformation engine to perform the transformations on the join points selected by the architect. We have three implementations for the processor, each of which uses a specific tool: AGG based on graph theory [102], [54], CIAO Prolog based on predicate logic [65], and DROOLS based on production rules [63].

The longer-term goal of TranSAT tools consists of comparing the different technologies used to implement an efficient transformation operator. Furthermore, we are going to experiment applying TranSAT's transformation process to other kinds of diagrams such as class diagrams or deployment diagrams. The work on TranSAT was a part of Olivier Barais's PhD thesis. It is mainly developed by Missi Tran-Anh and Olivier Barais.

SafArchie Studio and TranSAT tools are open source software available at http://transat.gforge.inria.fr

5.10. Spoon

Keywords: *Program transformation*.

Participants: Carlos Francisco Noguera Garcia, Renaud Pawlak [correspondant], Nicolas Petitprez.

Spoon [17], [50] is a project that started in March 2005 and was officially hosted by INRIA Gforge in September 2005. The goal of Spoon is to provide a core API and associated tools for static analysis and generative programming within the Java 5+ environment. Spoon must be seen as a basis to ensure Software Quality through code validation and generation. It can be used in the software development process during the validation phases, as well as for engineering or re-engineering software.

The first key point of Spoon is to provide a well-typed and comprehensive AST API which is designed to facilitate analysis and transformation work for programmers. Scanners and processors allow the programmer to implement various program traversal strategies on the Java program. Also, the program representation is built with a well known and well tested open source Java compiler: The Eclipse JDT compiler, which ensures the support of the latest Java features.

The second key point of Spoon is to provide a pure Java API to specify program transformations using a well-typed generative programming technique (called Spoon Templates). By using well-typed templates, Spoon makes programming of transformations easier and safer for the end-user programmers.

Finally, the third key point of Spoon is that it provides an Eclipse plugin (SpoonJDT) that allows the programmers to package validations and transformations into compilation components called Spoonlets. These components can be deployed in the Eclipse plugin to enhance the Java compiler in a seamless and well-integrated way. For example, thanks to Eclipse's incremental compilation, errors and warnings coming from Spoonlets are reported as regular compilation mistakes, along with the typing of the program. The fact that Spoonlet-defined errors can be reported as the programmer types in the code (exactly like spelling or grammar mistakes are reported real-time by modern text editors) is of primary importance to produce high-quality code. Indeed, it is a recognized fact that the exact moment a programmer introduces a defect in the program is also the best time to fix it - because it is the moment when the programmer has the best understanding of what has just been written.

Many projects and experiments have been conducted around Spoon in the Jacquard project, but also in other INRIA projects and outside of INRIA, by independent developers. A non-exhaustive list follows:

- Spoon-AOP: a work on implementing AOP with Spoon in the context of middleware [17].
- AVal: a work done within the PhD of Carlos Nogera and that consists of a framework for validating annotation sets forming DSLs. This work was published in [33].
- Fraclet: a work done in the context of Nicolas Pessemier's PhD, in collaboration with Romain Rouvoy, that defines and implements a Java annotation based DSL for Fractal, the ObjectWeb component model. This work was published in [40].
- AOKell: an alternative implementation of the ObjectWeb Fractal container using AOP and Spoon.
- Spoon-EMF: a work done by Olivier Barais (TRISKELL INRIA Project) to provide an EMF compliant implementation of the Spoon API.
- Spoon-JMX: a work done by Didier Donsez (Grenoble University) for automatically transforming Java programs to support JMX.
- JUnit Suite Maker: a project to control your JUnit tests with Spoon and started by an independent developer. See http://perso.orange.fr/peupeu/sw/java/junitsuitemaker/junitsuitemaker.html.
- VSuite: a validation suite for Java started by Renaud Pawlak and Nicolas Petitprez.

Spoon is open source software available at http://spoon.gforge.inria.fr

6. New Results

6.1. Open Middleware for the CCM

6.1.1. Generic Framework for Large Scale Distributed Deployment

Participants: Areski Flissi, Philippe Merle.

Deployment of distributed applications on large and complex systems, such as grid infrastructures or ubiquitous environments, becomes a more and more complex activity. Deployers/users spend a lot of time to prepare, install software, libraries or binaries on remote nodes, configure environments and middleware, start application servers, and eventually start their applications. The problem is that the deployment process is composed of many heterogeneous tasks that have to be orchestrated in a specific order. Indeed, dependencies and synchronization problems often exist between software or elementary deployment tasks. As a consequence, the automatization of the deployment process is currently very difficult to achieve.

To address this problem, we propose in [27] a generic deployment framework called FDF (Fractal Deployment Framework), which allows automation of the execution of heterogeneous tasks composing the whole deployment process. Our approach is based on the reification as software components of all required deployment mechanisms or existing tools involved in the deployment process, such as remote access and file transfer protocols, shells, user access information, node ports and hostnames. Software is also reified as particular components which are called personalities. Personalities are written once by developers for each kind of software to deploy. Bindings between components represent dependencies. These components are composed and assembled together and the obtained composite represents the final configuration to deploy. In other words, execution of the composite means execution of the deployment process. FDF automatically orchestrates the deployment process. FDF allows users to just describe the configuration to deploy in a natural language instead of programming or scripting how the deployment process is executed.

FDF is implemented using the Fractal component model and the Java programming language. It is independent of the technology of the software to deploy and its granularity. Personalities for many component-based platforms have been written such as OpenCCM, JOnAS, JBoss, OSGi or PeTALS servers. Furthermore, FDF has been successfully experimented for the deployment of OpenCCM middleware and CORBA component-based applications on one thousand nodes of Grid5000, the french grid infrastructure dedicated to computer science research.

6.1.2. Distributed Autonomous Component-based Architectures

Participants: Jérémy Dubus, Philippe Merle.

With the emergence of *open distributed environments* (ODE), such as *grid* and *ubiquitous* computing, the management of systems during their execution has become a new challenge. Machines appear and disappear in an unpredictable way, and applications deployed on these environments must adapt their structures to address these changes. IBM proposes *autonomic computing* for building systems able to manage autonomously one concern of their execution (*e.g.*, self-healing, self-sizing, self-optimization). This proposition relies on the core concept of a *control loop*, which consists of four classical phases: monitoring the system, analyzing changes monitored, preparing a reconfiguration to handle the change, and finally executing this change. All of these phases rely on the *knowledge* that encompasses all relevant information about the application. The *autonomic policies*, in charge of defining the way an application should self-adapt, are defined using this paradigm.

Our goal is to propose a framework with adequate mechanisms to build reliable autonomic component-based architectures. The result of our research work is called DACAR for Distributed Autonomous Component-based ARchitectures.

In [25], we have defined the core concepts of our approach. First, we would like the description of autonomic architecture to be generic and independent from the underlying middleware. Then, we propose to use the OMG specification called *Deployment and Configuration of Distributed Component-based Applications* (D&C). It provides generic concepts to express the structure of component-based applications. The *reified architecture* built using D&C can be then extended with some autonomic policies. We found that the Event-Condition-Action (ECA) rule paradigm is well tailored to specify fine-grained autonomic policies. At runtime, the system is reified and the autonomic policies expressed using ECA are applied to this representation. The reified architecture is causally linked to the real system using some dedicated ECA policies. In [24], we illustrate the feasibility of our ECA approach in a complete ubiquitous example.

In [23], we show that our reified architecture in fact fits well with the definition of a model. This is an abstract representation of a system in order to facilitate the management of one concern of the application. In our case the concern is about self-configuration (and self-deployment) of an application. We propose a modeling process to build self-configuring component-based applications based on D&C and ECA metamodels. However, these models are not classical models since they must be executed: they are runtime models that must be causally linked to the running system.

Current work aims at exploiting this modelisation in order to go deeper into building efficient autonomic policies. Indeed all other works focusing on autonomic applications propose to build a control loop in a programmative way, which is prone to error. Moreover, using programmation it is impossible to check the

execution behaviour of the set of autonomous policies together. Our approach based on models allows the building of autonomous systems in a more abstract way, without getting lost in implementation details. Moreover, providing adequate concepts in our DACAR metamodel should allow us to check statically the resulting models, and to validate the autonomic policies as well as their interaction. We are currently working on the definition of a metamodel of reliable autonomous component-based applications relying on statically validated autonomic policies.

The DACAR prototype has been tested with many examples using the OpenCCM platform as the component-based application execution support.

6.1.3. Component-Based Software Framework for Building Transaction Services

Keywords: *Component-Based Software Framework*, *GoTM*, *Middleware Transaction Services*.

Participants: Philippe Merle, Romain Rouvoy.

GoTM is a component-based software framework for building middleware transaction services.

6.1.3.1. Overview of the GoTM Activity

Transactions have always been involved in various applications since they have been introduced in databases. Many transaction services have been developed to address various transaction standards and various transaction models. Furthermore, these transaction services are more and more difficult to build since the complexity of the transaction standards is constantly increasing. Each transaction service implements pieces of code that have already been written in other transaction services. As a consequence, there is no code factorization between the transaction services and the added values of each transaction service, such as extensibility or performance, are never reused in another transaction service.

In [13] and [38], we present GoTM, a Component-Based Adaptive Middleware (CBAM) software framework. It can be used to build various transaction services that are compliant with existing transaction standards (OMG OTS, Sun JTS, etc.). GoTM provides adaptive properties to support different transaction models and standards in the same transaction service. GoTM also supports the definition of new transaction models and standards as new components of the framework. Finally, GoTM provides (re)configurability, extensibility and adaptability properties as added values. The implementation of the GoTM framework is based on the Fractal component model. The next sections illustrate two experiences performed this year with the GoTM framework.

6.1.3.2. Building Heterogeneous Transaction Services

The diversity of transaction services leads to compatibility problems among applications using different transaction standards. This compatibility usually ensures that transaction services can cooperate in a system. To deal with this issue, current trends use coordination protocols. Coordination protocols are responsible for synchronizing the execution of transaction services based on different transaction standards. Nevertheless, these protocols can be intrusive and often introduce an additional complexity into the system.

In [13] and [41], we present an approach to build an *Adapted Transaction Service*, called ATS, that supports several transaction standards concurrently. The objective of ATS is to make the transaction standards composition easier. To introduce ATS, we present how the *Object Transaction Service* (OTS), *Web Services Atomic Transaction* (WS-AT) and *Java Transaction Service* (JTS) standards can be composed. To achieve this, the OTS, WS-AT and JTS interfaces are analyzed and the required/provided functions are identified. The functions are specialized as strategies to implement transaction standard semantics. The resulting ATS is built by composition of these strategies and adapters. Adapters ensure the compliance with transaction standard interfaces. Besides, the ATS implementation is introduced, which is implemented with the GoTM framework and the Fractal component model.

We show that this approach does not introduce an additional overhead to legacy applications and supports scalability well. Moreover, this approach can be easily extended to support additional transaction standards. Future work will investigate the definition of personalities for Web Services Transaction and Activity Services.

6.1.3.3. Supporting Dynamic Adaptation of 2-Phase Commit Protocols

For 30 years, transactional protocols have been defined to address specific application needs. Traditionally, when implementing a transaction service, a protocol is chosen and it remains the same during the system execution. Nevertheless, the dynamic nature of current application contexts (e.g., mobile, ad-hoc, peer-to-peer) and behavioral variations (semantic-related aspects) motivate the need for application adaptation. Next generation system applications should be adaptive or even better self-adaptive. In [13] and [42], we propose (1) a component-based architecture of standard 2PC-based protocols designed using UML sequence diagrams and (2) a Context-Aware Transaction Service, named CATE. Self-adaptation is obtained by behaviour awareness and component-based reconfiguration. This allows CATE to select the most appropriate protocol according to the context.

We have shown that using CATE performs better than using only one commit protocol in a variable system and that the reconfiguration cost can be negligible. Future work will investigate the design of a dedicated high-level model to describe transaction validation protocols.

6.1.4. Leveraging Component-Oriented Programming Using Attribute-Oriented Programming

Keywords: *Attribute-Oriented Programming*, *Component-Oriented Programming*, *Fraclet*.

Participants: Philippe Merle, Renaud Pawlak, Nicolas Pessemier, Romain Rouvoy.

Fraclet is an attribute-oriented programming model for developing reliable components.

6.1.4.1. Leveraging Fractal-Based Developments

Component-Based Software Engineering (CBSE) is concerned with the development of highly reusable business components which declare contractually specified interfaces to communicate with each other. CBSE facilitates the development of high quality applications with shorter development cycles and reduces coding effort. However, in practice, it appears that the component developer's task is not only devoted to the design and implementation of the business logic of the applications but also to the integration of redundant and error-prone non-functional properties.

A convenient way to address this issue is to use *Attribute-Oriented Programming* (@OP) techniques. @OP proposes to mark program code with metadata to clearly separate the business logic from a domain-specific logic (typically non-functional properties). @OP is gaining popularity with the recent introduction of annotations in Java 2 standard edition (J2SE) 5.0, XDoclet, and attributes in C#. Recently, the Enterprise Java Bean (EJB) 3.0 specification extensively uses annotations to make EJB programming easier. The Service Component Architecture (SCA) component implementation model provides a series of annotations which can be placed in the code to mark significant elements of the implementation which are used by the SCA runtime.

In [13], [39], [40], we present *Fraclet*, an annotation-based framework using @OP to leverage Fractal component programming. Fraclet provides a set of dedicated annotations to mark the Fractal-related non-functional properties in the program code. To achieve this goal, Fraclet introduces a seamless design process for Fractal component developers. Thanks to @OP, most of the component artifacts are automatically generated.

Fraclet is developed to deal with the Java implementations of the Fractal component model. We have experimented with two different, but functionally equivalent, implementations of the Fraclet annotation framework. The first one uses XDoclet and Velocity to define the code generators and to produce the various artifacts required by the Fractal component model. The second one uses Spoon [17], a Java 5-compatible processing tool, which supports the processing of Java 5 annotations. The Fraclet developer can then take advantage of Java 5 type safety and annotations. Regardless of the implementations, we show that, using Fraclet, about 50% of the handwritten program code can be kept while the rest of the program code is automatically generated and continuously integrated, without losing the semantics of the application.

6.1.4.2. Supporting Various Component Models

Component-oriented programming has achieved wide acceptance in the domain of software engineering by improving productivity, reusability and composition. This success has also encouraged the emergence of a plethora of component models. Nevertheless, even if the abstract models of existing component models are quite similar, their programming models can differ a lot. The programming model applies these concepts to a particular programming language, while introducing some non-functional code specific to the component model. Thus, this *non-functional code* is tangled with the *business code* of the application. Furthermore, if the abstract models of existing component models are quite similar, their programming models can differ a lot. This drawback limits the reuse and composition of components implemented using different programming models.

In [13] and [37], we introduce a reification of an abstract model common to several component models. This reification is presented as an annotation framework, which allows the developer to annotate the program code with the elements of the abstract component model. Then, using a generator, the annotated program code is completed according to the programming model of the component model to be supported by the component runtime environment. This paper shows that this annotation framework provides a significant simplification of the program code by removing all dependencies on the component model interfaces. These benefits are illustrated with the OpenCOM and Fractal component models.

6.2. Aspect-Oriented Design of Dynamic Components Assemblies

6.2.1. Early Aspects and Model Driven Engineering

Participants: Maja D'Hondt, Laurence Duchien.

Model inconsistency management is a crucial aspect of model-driven software engineering. It is therefore important to provide automated support for this activity. The problem is, however, that the resolution of inconsistencies may give rise to new inconsistencies. To address this problem, we propose to express inconsistency detection and resolutions as graph transformation rules, and to apply the theory of critical pair analysis to analyse potential dependencies between the detection and resolution of model inconsistencies. As a proof-of-concept, we report on an experiment that we have carried out along these lines using the critical pair analysis algorithm implemented in the state-of-the-art graph transformation tool AGG. The results show that both anticipated and unexpected dependencies between inconsistency detection and resolution rules are found by AGG. We discuss how the integration of the proposed approach into contemporary modelling tools can improve inconsistency management in various ways [31], [47].

In a second work, we propose an approach that combines Model-Driven Engineering and Aspect-Oriented Software Development in order to automatically translate high-level business rules to aspects and integrate them with existing object-oriented applications. The separation of rule-based knowledge from the core application as explicit business rules has been the focus of many existing approaches. However, they fail at supporting rules that are both high-level, defined in domain terms, and operational, i.e. automatically executable from the core application. We propose high-level languages for expressing business rules at the domain level, as well as their connections to the core application. We provide support for automatically translating high-level rules to object-oriented programs and their connections to aspects, since these crosscut the core application. Separation of concerns is preserved at the domain and implementation levels, facilitating traceability, reusability and adaptability. A prototype implementation highlighting the challenges encountered in the transformations is presented [15], [20].

6.2.2. Aspects at Design Time

Participants: Laurence Duchien, Anne-Françoise Le Meur, Missi Tran-Anh, Guillaume Waignier.

SafArchie Studio has been developed in the context of Olivier Barais' PhD thesis [58] and aims to facilitate the construction, analysis and evolution of software architectures. One of the features of SafArchie Studio is to allow the architect to build architectures by incrementally adding new functionalities. This approach, inspired by Aspect-Oriented Programming, relies on both the definition of architecture integration patterns and the use of a weaver to integrate patterns into the target architecture. A pattern consists of three parts: a sub architecture, a join point mask and a set of transformation rules. The sub architecture is a component assembly corresponding to a given functionality. The join point mask expresses properties that the target architecture must satisfy for the integration to be possible. Finally, the set of transformation rules specifies the operations that the weaver has to perform to integrate the sub architecture into the target architecture.

The languages used to express the join point mask as well as the transformation rules have been specifically designed to enable verifications that ensure that the integration of a given concern, represented by the sub-architecture of a pattern, will be performed safely, *i.e.,* will result into a correct architecture. The various verifications include static verifications that check coherence properties once the pattern has been defined, and dynamic verifications that focus on the parts of the architecture that are affected by the pattern [18].

Several tools have been used to implement the integration process, which has been decomposed into three distinct stages. The first stage consists in performing static analysis on a newly created architectural pattern. The second step focuses on finding the integration sites on the target architecture, *i.e.,* the parts of the architecture that satisfy the properties expressed by a given join point mask. The last stage addresses the actual integration of the architectural pattern into the target architecture. These stages are implemented using different mechanisms: graph transformation, forward-chaining and backward-chaining. The actual tools used are (AGG), (Drools) and (Prolog). AGG, which is a graph transformation tool, and Drools, which is a forward-chaining rule engine, are both data-driven whereas Prolog, which is a backward-chaining rule engine, is goal-driven.

When AGG is used, the software architecture is seen as a graph. In this context, the detection of integration sites relies on the graph matching feature provided by AGG. Moreover the transformation rules applied on the architecture are translated into a set of transformation rules to be applied on its associated graph. In the case of Drools and Prolog, the elements of the software architecture are put in a knowledge base. The detection of integration sites and the verification of architectural patterns with Prolog are based on queries whereas they rely on pattern matches and rule activations with Drools. The transformations are performed by modifying the knowledge base in both cases [46]. This implementation work has mainly been performed by Hanh-Missi Tran, and Olivier Barais. See http://transat.gforge.inria.fr.

This integration approach had been developed specifically to target a single ADL, *i.e.,* SafArchie. In the context of Guillaume Waignier's Master thesis [53], we have investigated the possibility of having a more generic approach, which has led to the development of FIESTA, a Framework for Incremental Evolution of SofTware Architectures. To generalize the SafArchie approach in order to be independent of any specific ADL, we have performed a domain analysis to understand the integration process in architectures described in various ADLs. This analysis has allowed us to identify the common architecture elements that are involved in integration, leading to the definition of a generic ADL model. This analysis has also enabled us to define more abstract expressions to specify a join point mask and the transformation rules. Furthermore, we have built our generic framework so that adding support for new ADLs is easy as the generic engine can be configured through the specification of a limited number of well-identified ADL-specific functions [48].

Finally, in the domain of soft real-time application design, the gap between component-specification models and the implementations often implies that the implementations cannot fully take advantage of the specification models. To limit this gap, we propose an approach to generate a QoS monitor from the timed behavior specification. To support this approach, we rely on two different component models: one focused on formal description and the other on practical implementation. Those models are interconnected by model transformation, using a Model-Driven Engineering style [43].

6.2.3. Logic Pointcut Languages for Object-Oriented Programs

Participants: Johan Brichau, Laurence Duchien, Carlos Francisco Noguera Garcia.

A crosscutting concern affects a software application in many heterogeneous places. An aspect that encapsulates such a crosscutting concern has a means for denoting these places in a pointcut expression. There have been several aspect-oriented approaches that use Prolog as a language for expressing pointcuts as logical queries. The first such approach, Carma [71] was developed at Prog-Ssel. More recently, other approaches have started using logic pointcut languages, such as LogicAJ [77], showing that this is a valid approach.

Johan Brichau joined the INRIA from January 1st to September 30th 2006 and has continued his work with Kim Mens (UCL) and Andy Kellens (VUB) on the fragile pointcut problem in aspect-oriented programming languages. Their joint research focuses on the use of an advanced (logic-based) pointcut language in an integration with the intensional views approach. Following the successful publication of their work at ECOOP 2006 in Nantes [29], they have met during a full week at the INRIA to improve their approach and execute more advanced experiments. In particular, they are extending the intensional views approach with tuple calculus to cope with the full predicate logic of the advanced pointcut language [28]. Furthermore, they have built a more advanced prototype and written a publication to the academic track of the European Smalltalk Conference [19].

In a second collaboration between Johan Brichau, Coen de Roover, Théo D'Hondt, Laurence Duchien and Carlos Noguera, the expertise of Carlos on templates [34] and Coen's static analysis and program investigation [22] expertises are combined into the development of a template language for the querying of programs based on static as well as dynamic properties. This work will permit the querying of the source code of a program by providing an example of the intended behaviour. This kind of querying has applications in pointcut languages, program comprehension, and design pattern mining, among others.

6.2.4. Aspects and Components for Software Architectures

Participants: Laurence Duchien, Guillaume Dufrêne, Frédéric Loiret, Nicolas Pessemier, Ales Plsek, Lionel Seinturier.

This action is concerned with the definition and the implementation of a programming environment for complex applications in the domain of middleware, embedded systems and ubiquitous computing. This action investigates the use of two software engineering techniques: component-based software engineering (CBSE) and aspect-oriented software development (AOSD).

In [35] and [36], we show that far from being conflicting, AOSD and CBSE can be integrated in an unified programming environment. The novelty of the approach is to give developers tools for building the business part of their applications with software components and the technical part with aspects. An isomorphism has been defined between components and aspects: aspects are components and are connected to the rest of the software architecture with bindings whose semantics has been defined. This model has been used for implementing applications in various case studies such as the steering of numerical applications (in the context of the ARC COA action).

In [44] and [45], we propose an approach for designing and implementing an adaptable component framework. The idea is to let the non-functional services which are provided by the framework to components be able to change. This leads to a component framework which can be adapted depending on the properties of the execution context or on the user requirements. The novelty of the approach is that we are using aspects to achieve this goal, i.e., the adaptability of the component framework is provided by aspects. The second scientific contribution is that this is a reflective framework in the sense that the notion of a component is used both for the applications which are hosted by the framework and for the implementation of the framework itself.

These two results were achieved partially in the context of the ongoing grant with France Telecom R&D. Two open source platforms have been released in this context: FAC and AOKell. AOKell is hosted by the ObjectWeb consortium for open source middleware. It takes the form of an implementation of the Fractal specification and is thus released as a subproject of the Fractal project.

Two other results in the domain of AOSD and CBSE are worth noticing. Firstly, concerning AOSD, we conducted a study in the context of the LAFMI AProSec project where we proposed an aspect-based model and its implementation for detecting malicious attacks (of the form SQL-injection and cross-side scripting)

on Web applications. We showed that the use of aspects for securing such applications is less intrusive than traditional techniques and does not introduce any performance penalties. Secondly, concerning CBSE, we set up a bridge between the Fractal and the SCA component models in the context of the master thesis of Guillaume Dufrêne [51]. We showed that, far from being incompatible, these two models can interoperate. This experiment opens many interesting perspectives for using software components at different levels of granularity in service-oriented architectures.

6.2.5. Complex Aspects

Participants: Denis Conan, Johan Fabry, Nicolas Pessemier, Lionel Seinturier.

This topic focuses on the software engineering properties of the aspect itself. As the aspect itself is a piece of software, it should obey the software engineering principles of good modularisation and separation of concerns. Instead we see that, aspects themselves may suffer from bad modularisation and lack adequate separation of concerns. In some cases this leads to aspect code which itself is tangled, a phenomenon we call tangled aspect code.

This topic on complex aspects has been largely focused on re-examining previous work to obtain new insights concerning software engineering issues due to the complexity of the aspect. This previous work consists of two separate elements, both in the domain of distributed systems. The first element is the domain-specific aspect language KALA for advanced transaction management. The second is a case-study of the use of the JAC language to aspectize the use of group communication systems. In both cases problems were encountered regarding the modularization of the aspect itself leading to tangled aspect code [26]. In other words, we were not able to cleanly separate the aspect itself into a collection of modules. In studying these two cases we gained new knowledge as to the fundamental underlying cause, and a possible solution strategy. The fundamental underlying cause of the problems with regard to modularity of the aspect itself, can be concisely stated as follows: "The aspect itself is also subject to a dominant decomposition, which leads to the emergence of cross-cutting sub-concerns, i.e., sub-aspects.". With a large aspect, splitting it into sub-concerns will yield a dominant decomposition. Combined with the known phenomenon of the 'tyranny of the dominant decomposition', this yields that in a large aspect unavoidably cross-cutting sub-concerns will arise. Modularizing these will require sub-aspects. In both cases introduced above, we encounter this as follows: In KALA the dominant decomposition imposed by the language is the life-cycle of a transaction in begin, commit and abort phases. The different modules however encapsulate sub-concerns of the implementation of an advanced transaction model and these impact multiple phases of the life-cycle. In group communication systems two possible dominant decompositions can be made. These are the, so called, vertical stack or horizontal stack. Remarkably, in both decompositions the different sub-concerns implementing the particularities of the protocols used are cross-cutting. This shows that choosing a different decomposition does not necessarily yield the absence of tangled aspect code. We have investigated possible solution strategies, based on the following requirement: To avoid tangled aspect code we require more composition support at the level of the aspect. Current popular aspect languages typically provide aspect precedence as the only composition mechanism, which is inadequate to solve the problem of tangled aspect code. To achieve more composition support we propose the use of more symmetrical aspect paradigms. With such symmetry, the code of the aspect itself is treated as any other module, and therefore all composition functionality at the level of the base code can be used, including aspects themselves.

6.2.6. Transformation Languages and Tools

Participants: Carlos Francisco Noguera Garcia, Renaud Pawlak, Nicolas Petitprez.

In order to support analysis and design-level validation in an efficient way, we have worked at building language-level transformation tools. For this, we have developed Spoon [17], which is an annotation-driven program transformation tool. In 2004, Sun Microsystems released Java 5, which includes "generics" and "annotations". With annotations, the Java 5 programmers can define and attach metadata on program elements. These metadata allow for declarative configuration of the programs to define non-functional properties. In many cases, an annotation-based configuration is preferred over a typically XML-based one, because it is more integrated into Java, much safer (type-safety), and it avoids information redundancy. However, current tools

to deal with annotations are limited to large-grained annotation-based program validation and documentation generation. Spoon's goal is to answer the needs of the industry and of the Java community in general for a tool that fully supports generics and annotations and that allows for fined-grained and type-safe program validation and transformations. Spoon has applications in several domains. Spoon can be used in component containers (to implement annotation-driven deployment), for general software engineering (in particular AOSD can benefit from the use of annotation to parameterize the pointcuts), ubiquitous computing (annotations would help the programs to target the right environment), real-time systems (programmers would then define time constraints annotations directly in the Java code), etc.

We also study and develop on attribute-oriented programming (@OP) and expressive pointcut languages. For @OP, he defined AVal: an @DSL (Attribute-based Domain Specific Language) validator on the top of Spoon and distributed as a Spoonlet. The idea behind AVal consists of defining meta-attributes (attributes for attributes), which constrain the @DSL and avoid misusing it. AVal has been used to validate the Java 5 Fraclet @DSL for Fractal. This work has been published in [33]. From the expressive pointcut languages, Carlos has been working on matching program structures and behaviors using code templates. He used Spoon templates to express advanced pointcuts in Spoon AOP.

7. Contracts and Grants with Industry

7.1. France Telecom

Participants: Laurence Duchien, Nicolas Pessemier, Lionel Seinturier.

This contract is a CRE ("Contrat de Recherche Externe") that takes place in the context of the "accord-cadre" between INRIA and France Telecom R&D. This is a 3-year contract that begun in October 2004. The scientific teams involved in the project are for INRIA, the Jacquard project-team, and for France Telecom R&D, the ASR/Polair department. The contract goal is to study and construct component- and aspect-based software architectures. The Fractal component model from France Telecom R&D and the JAC AOP framework from Jacquard form the background of this work. The expected result is a model (FAC for Fractal Aspect Component) that merges and unifies aspects and components. The PhD thesis work of Nicolas Pessemier is directly related to this contract.

7.2. NorSys

Participants: Dolorès Diaz, Laurence Duchien, Lionel Seinturier.

This contract is associated to a CIFRE PhD thesis between the Jacquard project-team and the NorSys service company. The goal of the contract is to study aspect-orientation in the early stages of software development. AOP emerged as a programming technique but the question is now open in the international research community to tell whether it can also bring to innovations into the early stages of requirement engineering, analysis and design. This contract begun in January 2004. The PhD thesis work of Dolores Diaz is directly related to this contract.

8. Other Grants and Activities

8.1. Regional Initiatives

8.1.1. MOSAIQUES

Participant: All members of the Jacquard project.

The MOSAIQUES Project ("MOdèles et InfraStructures pour Applications ubIQUitairES" or Models and middleware for ubiquitous applications) defines a design and programming framework for application definitions that run in an ubiquitous environment. The project includes the University of Lille with LIFL Laboratory (STC and SMAC teams) and INRIA projects Jacquard and POPS, TRIGONE laboratory, INRETS, Ecole des Mines de Douai and the University of Valenciennes and of Hainaut-Cambrésis. Application domains are transportation and e-learning systems. Laurence Duchien is in charge of this project.

8.2. National Initiatives

8.2.1. ARA REVE

Participants: Laurence Duchien, Anne-Françoise Le Meur, Frédéric Loiret, Ales Plsek, Lionel Seinturier.

REVE (safe Reuse of Embedded components in heterogeneous enVironmEnts) is a 3-year project funded by the ARA SI (Sécurité, Systèmes embarqués et Intelligence Ambiante) program of the ANR. Five partners are involved: CEA, CNAM, INRETS, INRIA and INSA. The objective of the project is to define a component model, a type system and an execution platform for context-aware embedded applications [30]. Lionel Seinturier is the scientific leader of this project.

8.2.2. ARC COA

Participants: Philippe Merle, Lionel Seinturier, Renaud Pawlak, Nicolas Pessemier.

This 2-year project is funded by the INRIA Cooperative Research Initiative (ARC) whose partners are the PARIS, SCALAPLIX and Jacquard Project-Teams. The objective is to design an experimental platform allowing dynamic adaptation and steering of distributed numerical simulation applications using aspect weaving techniques on top of component models.

8.2.3. RNTL FAROS

Participants: Laurence Duchien, Guillaume Dufrêne, Anne-Françoise Le Meur, Lionel Seinturier, Guillaume Waignier.

FAROS (composition de contrats pour la Fiabilité d'ARchitectures Orientées Services) is a RNTL project involving EDF R&D, France Télécom, Alicante, IRISA, I3S and LIFL. This project addresses issues related to the safe integration of services in service-oriented architectures. The overall goal of this project is to provide a methodology to guide and automate the different tasks involved in the integration process, thus enabling the integration process to become reproducible. Our approach will be based on the definition of contracts, which will allow static and dynamic verifications to be performed. The feasibility of our approach will be demonstrated through the development of three dynamic and constrained applications (Health, Education, Electricity).

The project has started in January 2006 and is scheduled for a 36-months period. A first deliverable, a state of the art on contracts and service composition, has been produced [49].

8.2.4. RNTL JOnES

Participants: Nicolas Dolet, Philippe Merle, Romain Rouvoy.

JOnES is a 24-months ANR RNTL project started in January 2006 and involving INRIA (ObjectWeb, Sardes and Jacquard teams), EBM WebSourcing, ENSTIMAC, France Telecom R&D, Open Wide, and ScalAgent Distributed Technologies.

The goal of this project is to design and develop an open source distributed Enterprise Software Bus (ESB) infrastructure based on the Java Business Integration (JBI) specification and implemented with the Fractal component model. The major innovation of JOnES is to propose a new distributed open framework for ESB based on the interconnection of JBI-compliant containers. The project results are distributed through the ObjectWeb PEtALS open source project (see http://petals.objectweb.org).

The main Jacquard contributions are: Fraclet as the Fractal programming model used to build PEtALS components, Fractal Deployment Framework (FDF) to deploy distributed PEtALS infrastructures and applications, and GoTM to build the PEtALS distributed transaction service.

8.3. European Initiatives

8.3.1. AOSD-Europe

Participants: Laurence Duchien, Lionel Seinturier, Renaud Pawlak, Carlos Francisco Noguera Garcia.

AOSD-Europe is a Network of Excellence (NoE) in aspect-oriented software development within IST-FP6. It joins 11 research groups and among them members of the Jacquard project and other members from OBASCO, Pop-Art and Triskell INRIA projects. The NoE is led by Lancaster University, Darmstadt University and University of Twente. The goal of the NoE is to harmonise, integrate and strengthen European research activities on all issues related to aspect orientation: analysis, design, development, formalization, applications, empirical studies.

8.3.2. CALA

Participants: Johan Brichau, Maja D'Hondt, Laurence Duchien, Johan Fabry, Anne-Françoise Le Meur, Carlos Francisco Noguera Garcia, Nicolas Pessemier, Lionel Seinturier.

The INRIA project Jacquard together with the Prog-Ssel team focus on crosscutting concerns at the level of software architectures, component models and object-oriented programs. More specifically, we work together in CALA (Aspect-Oriented Software Development in Languages, Component Models and Architectures), an INRIA Associate Team project, (http://jacquard.lifl.fr/software/VUB/lifl-vub.html) for elaborating and extending joint work we have already started in the field of AOSD on four particular topics: (1) logic pointcuts at the object-oriented programming level, (2) complex aspects, (3) the unification of aspects and components and (4) pointcuts at the architectural level.

8.3.3. ITEA S4ALL

Participants: Nicolas Dolet, James Manley, Philippe Merle.

S4ALL ('Services-for-All') is an ITEA project whose partners are Alcatel CIT, Bull, Capricode, Fraunhofer Fokus, HIIT, INRIA, Instituto de Telecomunicaçoes, INT, mCENTRIC, Nokia, PT Inovaçao, Schneider Electric, Thales, Université Joseph Fourier, Universidad Politécnica de Madrid, University of Kassel, Vodafone, and Xquark/Odonata. The main vision is to build "A world of user-centric services that are easy to create, share and use".

Our contribution in this project is to design and build a component-based framework to automatically deploy any kind of middleware and business components like J2EE components, OSGi services, and Web services.

8.4. International Initiative

8.4.1. LAFMI AProSec

Participants: Laurence Duchien, Lionel Seinturier.

AProSec (Aspects for Programming Secure distributed systems) is a franco-mexican LAFMI project between INRIA Jacquard and Dr Gomez's team from Instituto Tecnologico de Monterrey (ITESM-CEM) located in Mexico City. The goal of the project is to study advanced software engineering techniques based on aspect-oriented programming for securing Web applications.

8.4.2. ICT-Asia

Participant: Philippe Merle.

This is a French-Asian cooperation on open adaptive middleware for ubiquitous environments. Partners are National University of Defense Technology (China), Open Source Software Resource Center (Vietnam), Peking University (China), and INRIA (Sardes, ObjectWeb and Jacquard project teams).

It is widely anticipated that future ubiquitous computing environments will be highly dynamic, subject to constant changes and of ever-increasing complexity. This in turn motivates the construction of dynamically configurable software infrastructures to provide a consistent, systematic basis for system evolution, control and management.

Targeted research is required into autonomic and complex adaptive approaches to distributed system design, along with non-intrusive middleware virtualization techniques. Such research will provide the essential foundations upon which distributed, adaptive, self-managing, self-healing systems, capable of transparently supporting the growing needs of distributed e-business, e-gov, e-learning applications, will be built.

8.4.3. ObjectWeb

Participants: Nicolas Dolet, Guillaume Dufrêne, James Manley, Philippe Merle, Nicolas Pessemier, Lionel Seinturier, Romain Rouvoy, Renaud Pawlak.

ObjectWeb is an international consortium to promote high quality open source middleware. The vision of ObjectWeb is that of a set of components which can be assembled to offer high quality middleware. We are member of this consortium and Philippe Merle is member of the ObjectWeb College of Architects. Our software projects AOKell, Fraclet, Fractal Explorer, GoTM, JAC, and OpenCCM are hosted by the consortium.

8.4.4. OMG

Participant: Philippe Merle.

We work in the international consortium Object Management Group (OMG) since 1997. OMG defines well-known standards: CORBA, UML, MOF, MDA. We can quote our contributions to the OMG standardization work with the CorbaScript language (proposed to the Scripting Language for CORBA RFP, and accepted as the CORBA Scripting Language chapter of CORBA 3.x) and with the CORBA Component Model (CCM) chapter for which we lead the response group and the revision task force. We also participated in the definition of a UML profile for CORBA Components.

Philippe Merle is currently:

- Official representant of the INRIA at OMG,
- Member of the OMG Deployment Revision Task Force (RTF),
- Member of the OMG QoS for CORBA Components Revision Task Force (RTF),
- Member of the OMG Streams for CCM Revision Task Force (RTF).

9. Dissemination

9.1. Scientific Community Animation

9.1.1. Examination Committees

- **Johan Brichau** was in the examination committee of the following PhD thesis:
  - Dirk Deridder, 2006, Vrije Universiteit Brussel, Belgium (co-advisor)
- **Maja D'Hondt** was in the examination committee of the following PhD thesis:
  - E. Van Paesschen, July 2006, Vrije Universiteit Brussel, Belgium (co-advisor)
- **Laurence Duchien** was in the examination committee of the following PhD thesis:
  - H. Mcheick, April 2006, University of Montréal (referee)
  - A. Ocello, June 2006, University of Nice (chair)
  - R. Santos Marquez, September 2006, Institut National Polytechnique de Lorraine (referee)

  - C. Deleray, October 2006, University of Marne-la-Vallée (chair)
  - C. Tibermacine, October 2006, University of South Britany (referee)
  - D. Durand, November 2006, University of Amiens (member)
  - N. Belhanafi Behlouli, November 2006, Institut National des Télécommunications (referee)
  - K. Guennoun, December 2006, University of Toulouse III (referee)
  - HDR J-M. Bruel, December 2006, University of Pau (referee)
  - H. Fakih, December 2006, University of Lille 1 (advisor)
  - J. Klein, December 2006, University of Rennes 1 (referee)
- **Jean-Marc Geib** was in the examination committee of the following PhD thesis:
  - Bassem Kosayba, April 2006, University of Lille (co-advisor)
  - HDR Manuel Davy, May 2006, Ecole Centrale de Lille (chair)
  - Alexis Muller, June 2006, University of Lille (co-advisor)
  - Vincent Benony, June 2006, University of Lille (co-advisor)
  - François Ingelrest, June 2006, University of Lille (chair)
  - HDR Thierry Villemeur, September 2006, University of Toulouse (referee)
  - Alexandre Courbot, September 2006, University of Lille (chair)
  - Tarik Filali Ansary, October 2006, University of Lille (chair)
  - HDR Christian Pérez, November 2006, University of Rennes (referee)
  - HDR Pierre-Alain Muller, November 2006, University of Rennes (referee)
  - Romain Rouvoy, December 2006, University of Lille (co-advisor)
- **Philippe Merle** was in the examination committee of the following PhD thesis:
  - Romain Rouvoy, December 2006, University of Lille (co-advisor)
- **Lionel Seinturier** was in the examination committee of the following PhD thesis:
  - Takoua Abdellatif, September 2006, Institut National Polytechnique de Grenoble (referee)
  - Christophe Deleray, October 2006, University of Marne-la-Vallée (referee)
  - David Durand, November 2006, University of Amiens (referee)
  - Assia Hachichi, December 2006, University of Paris 6 (referee)
  - Nicolas Salatgé, December 2006, Institut National Polytechnique de Toulouse (referee)
  - Iyad Alshaboni, December 2006, University of Lille (chair)

9.1.2. Journals, Conferences, Workshop

- **Johan Brichau** has been a member of the following committees:
  - PC-member of Net.Objectdays: Objects, Aspects, Services, the Web (NODe) track, 2006,
  - PC-member of Workshop on Domain-Specific Aspect Languages at GPCE, 2006,
  - PC-member of the International ERCIM Workshop on Software Evolution, April 2006, Lille, France,
  - Reviewer for IEEE Transactions on Aspect-Oriented Software Development (TAOSD),
  - Co-organisor of Software Properties for Languages and Aspect Technologies (SPLAT) workshop, AOSD 2006, Bonn, Germany,
  - Main organisor of Open Aspect Languages Workshop, AOSD 2006, Bonn, Germany,

  - Co-organisor of 1st Summer School on Aspect-Oriented Software Development, Vrije Universiteit Brussel – Belgium, from 10 to 14 July 2006.

- **Maja D'Hondt** has been a member of the following committees:
  - Program committee of AOSD 2006, Bonn, Germany, March 21, 2006,
  - Program committee of The Ninth ASTReNet Workshop, IEEE International Astrenet Aspect Analysis (AAA) Workshop, 24th October 2006, Benevento, Italy, Held in conjunction with 13th Working Conference in Reverse Engineering (WCRE 2006),
  - Program committee of the 3rd European Workshop on Aspects in Software (EWAS 2006), University of Twente, Enschede, The Netherlands, August 31th 2006,
  - Program committee of the workshop on Software Engineering Properties of Languages and Aspect Technologies (SPLAT 2006), workshop affiliated with AOSD 2006, Bonn, Germany, March 21th 2006,
  - Student Co-Chair with Yvonne Coady at AOSD 2006 (60 participants),
  - Organizing committee of International ERCIM workshop on Software Evolution, LIFL and INRIA, Université des Sciences et Technologies de Lille, France, Thursday 6 April and Friday 7 April 2006 (50 participants) [11].

- **Laurence Duchien** has been a member of the following committees:
  - Program committee of the national conference MCETECH, Montréal, May 2006,
  - Program chair of the French conference IDM (Ingénierie Dirigée par les Modèles), Lille, June 2006,
  - Program committee of the French workshop on Software Evolution, Nimes, March 2006,
  - Program committee of the international workshop ERCIM Software Evolution, Lille, April 2006,
  - Program committee of the French conference on operating systems (CFSE), Perpignan, October 2006,
  - Program committee of the first French conference on Software Architecture, Nantes, September 2006,
  - Program committee of the international conference FORTE - 26th IFIP WG 6.1 International Conference on Formal Methods for Networked and Distributed Systems, Paris, France, September 26-29 2006,
  - Program committee of special issue software evolution journal, RSTI-L'Objet, Hermès, March 2006,
  - Program committee of special issue "Views, points of view and roles: from concept to its using", RSTI-L'Objet, Hermès, December 2006,
  - Organizing committee of International ERCIM workshop on Software Evolution, LIFL, INRIA, Université des Sciences et Technologies de Lille, France, Thursday 6 April and Friday 7 April 2006 (50 participants) [11],
  - Organizing committee of the french conference IDM, Université des Sciences et Technologies de Lille, France, June 26-28 2006 (100 participants) [12].

- **Johan Fabry** has been a member of the following committees:
  - Organizing committee and program committee of DSAL'06 - workshop on Domain Specific Aspect Languages organized at GPCE 06 (International Conference on Generative Programming and Component Engineering - Sponsored by ACM SIGPLAN, in cooperation with ACM SIGSOFT), Portland, Oregon, October 22-26, 2006,

  - Editor of the special issue on Dependencies and Interactions with Aspects of the journal Transactions on Aspect-Oriented Programming (Springer Verlag).

- **Jean-Marc Geib** has been a member of the following committees:
  - Program committee of the french NOTERE 2006 conference,
  - Editorial board of the journal RSTI-L'Objet, Hermès.

- **Anne-Françoise Le Meur** has been a member of the following committees:
  - Program committee of SET'06 - IFIP Working Conference on Software Engineering Techniques, October 17-20, 2006, Warsaw, Poland,
  - Organizing committee and program committee of DSAL'06 - workshop on Domain Specific Aspect Languages organized at GPCE 06 (International Conference on Generative Programming and Component Engineering - Sponsored by ACM SIGPLAN, in cooperation with ACM SIGSOFT), October 22-26, 2006, Portland, Oregon.

- **Philippe Merle** has been a member of the following committees:
  - Program committee of the french conference Langages et Modèles à Objets, Nîmes, France, March 2006,
  - Program committee of the french 2ième Journées Multi-Agent et Composant (JMAC), Nîmes, France, March 2006,
  - Program committee of the French conference on operating systems (CFSE), Perpignan, October 2006,
  - Program committee of the french 5ème Journées Composants, Perpignan, France, October 2006,
  - Editorial board of the journal RSTI-L'Objet, Hermès.

- **Romain Rouvoy** has been a member of the following committees:
  - External referee for MCWC 2006, Mobile Computing and Wireless Communications International Conference, Amman - Jordan, September 2006,
  - Program committee of Middleware Doctoral Symposium, Melbourne, Australia, November 2006.

- **Lionel Seinturier** has been member of the following committees:
  - Program chair, 5ème Journées Composants, Perpignan, France, October 2006,
  - Program committee, 5th International Symposium on Software Composition, Vienna, Austria, March 2006,
  - Program committee, Langages et Modèles à Objets, Nîmes, France, March 2006,
  - Editorial board journal TSI (Technique et Science Informatiques),
  - Program committee, special issue on Adaptation and Context Management of the ISI (Ingénierie des Systèmes d'Information) journal.

- **Miscellaneous**
  - **Lionel Seinturier** gave a lecture at the 5th summer school on middleware and distributed applications (ICAR 2006). The topic of the lecture was on the Fractal component model and its associated development platforms.
  - The team has organized its internal seminar, Namur, Belgium, April 10-12th 2006.
  - The team has organized the CALA seminar (VUB and Jacquard), Lille, France, November 16-17th 2006.

9.2. Teaching

- **Jean-Marc Geib** teaches Object Oriented Design and Programming and Distributed Application Design in L3 and M1 at USTL, UFR IEEA.
- **Laurence Duchien** taught several courses until August 2006: Distributed Applications Design - Master Professionnel Sciences et Technologies Mention Informatique - M1 and Master Professionnel Sciences et Technologies Mention Informatique - M2 - Spécialité IAGL et TIIR at USTL, UFR IEEA. She was in charge of the Master Professionnel Sciences et Technologies Mention Informatique - M2 - Spécialité IAGL at USTL, UFR IEEA until September 2006.
- **Anne-Françoise Le Meur** teaches distributed application design (Master 1), databases and the internet (Master 2 pro), systems programming (licence).
- **Lionel Seinturier** teaches middleware (Master 1 and 2), component-based software engineering (Master 2), aspect-oriented programming (Master 2), and object-oriented design (Master 1).

Jacquard team members participate to some Research Masters of Computer Science (University of Lille, University of Montpellier, University of Valenciennes, ESSI in Nice and RPI University, Hartford, US) on the CCM, MDE and on AOP.

9.3. Administrative Responsibilities

- **Jean-Marc Geib** is the head of the LIFL laboratory (UMR 8022 CNRS-LIFL), the head of the LIFL CIM axis, and the head of the LIFL GOAL « Génie des Objets et Composants » team. He is member of the UFR board at USTL, member of CSE 27nd section of Universities of Lille 1, Lille 2 and Littoral, member of the orientation council of Télécom Lille 1, and member of the scientific committee of UR INRIA Futurs. He is the chair of the management committee TAC (Technologies Avancées de la Communication) of the Etat-Region Contract Nord-Pas-de-Calais and the coordinator of the communication program since 2004. He is member of the board committee and cofounder of IRCICA (Institut de Recherche sur les Composants matériels et logiciels pour l'Information et la Communication Avancée) that is a research federation between LIFL (Science computing), IEMN (electronics) and PhLAM (photonique). He is chair of the evaluation committee of the LIG laboratory at Grenoble, member of the evaluation committee of the CITI laboratory at Lyon, and member of the appelation commitee of INT at Evry. He is in charge of the RTP Distributed Systems of the CNRS STIC Dept. He is chair of the Réseau Thématique Prioritaire (RTP5) « Distributed Systems » of the CNRS STIC Department. He is expert for MSTP DS STIC. He is chair of the "Thèmatique 4" committee for the RNTL program of the french research agency (Agence Nationale de la Recherche). He is member of the executive committee of the french-mexican LAFMIA laboratory.
- **Laurence Duchien** is member of the UFR board at USTL, member of the LIFL scientific board, member of CSE 27nd section of Universities of Lille 1, CNAM and Paris 6. She is member of the scientific committee of the national ACI Security. She is member of ERCIM software evolution group. She is chair of the Scientific Commission of the UR INRIA-Futurs and Co-chair (with Jean-Louis Giavitto) of the Languages and Verification group of the GDR-CNRS GPL (Génie de la Programmation et du Logiciel). She was a member of the 2006 Selection Committee for the Junior Researcher permanent positions at IRISA Rennes.
- **Areski Flissi** is member of the "Comité des Utilisateurs des Moyens Informatiques" (CUMI) of UR Futurs and representing Jacquard members.
- **Philippe Merle** is in charge of the European Affairs within the Department for European and International Relations (DREI) for INRIA UR Futurs and is member of the Incitative Action Working Group (GTAI) of the Scientific and Technological Orientation Council (COST) of INRIA. He is member of the steering committee of the "Grilles, Système et Parallélisme" (GSP) working group of the CNRS ARS GdR.

- **Lionel Seinturier** was member of CSE 27 at the University of Nanterre and at CNAM until June 2006. He was member of Conseil National des Universités 27ème section until June 2006.
- **Romain Rouvoy** is member of the "Comité d'Unité de Recherche" (CUR) of UR Futurs and representing Ph. D. students.

10. Bibliography

Major publications by the team in recent years

[1] O. BARAIS, L. DUCHIEN, A.-F. LE MEUR. *A Framework to Specify Incremental Software Architecture Transformations*, in "31st EUROMICRO CONFERENCE on Software Engineering and Advanced Applications (SEAA 2005)", IEEE Computer Society, September 2005.

[2] C. DEMAREY, G. HARBONNIER, R. ROUVOY, P. MERLE. *Benchmarking the Round-Trip Latency of Various Java-Based Middleware Platforms*, in "Studia Informatica Universalis Regular Issue", ISBN: 2-912590-31-0, vol. 4, n^{o} 1, May 2005, p. 7-24.

[3] A. FLISSI, C. GRANSART, P. MERLE. *A Component-based Software Infrastructure for Ubiquitous Computing*, in "Proceedings of the 4th International Symposium on Parallel and Distributed Computing (ISPDC 2005), Lille, France", ISBN: 0-7695-2434-6, IEEE, July 2005, p. 183-190.

[4] T. MENS, R. VAN DER STRAETEN, M. D'HONDT. *Detecting and Resolving Model Inconsistencies Using Transformation Dependency Analysis*, in "Proceedings of the 9th International Conference on Model Driven Engineering Languages and Systems (MoDELS'06), Genova, Italy", Lecture Notes in Computer Science, vol. 4199, Springer, October 2006, p. 200–214.

[5] R. PAWLAK. *Spoon: Compile-time Annotation Processing for Middleware*, in "IEEE Distributed Systems Online", vol. 7, n^{o} 11, November 2006.

[6] R. PAWLAK, J.-P. RETAILLÉ, L. SEINTURIER. *Foundations of AOP for J2EE Development*, ISBN: 1-59059-507-6, APress, September 2005.

[7] R. PAWLAK, L. SEINTURIER, L. DUCHIEN, G. FLORIN, F. LEGOND-AUBRY, L. MARTELLI. *JAC : An Aspect-based Distributed Dynamic Framework*, in "Software Practice and Experience (SPE)", vol. 34, n^{o} 12, October 2004, p. 1119-1148.

[8] N. PESSEMIER, L. SEINTURIER, T. COUPAYE, L. DUCHIEN. *A Model for Developing Component-based and Aspect-oriented Systems*, in "Proceedings of the 5th International Symposium on Software Composition (SC'06), Vienna, Austria", Lecture Notes in Computer Science, vol. 4089, Springer-Verlag, March 2006, p. 259–273.

[9] R. ROUVOY, P. SERRANO-ALVARADO, P. MERLE. *Towards Context-Aware Transaction Services*, in "Proceedings of the 6th International Conference on Distributed Applications and Interoperable Systems (DAIS'06), Bologna, Italy", Lecture Notes in Computer Science, vol. 4025, Springer-Verlag, June 2006, p. 272–288.

[10] L. SEINTURIER, N. PESSEMIER, L. DUCHIEN, T. COUPAYE. *A Component Model Engineered with Components and Aspects*, in "Proceedings of the 9th International SIGSOFT Symposium on Component-Based Software Engineering (CBSE'06), Västeras, Sweden", Lecture Notes in Computer Science, vol. 4063, Springer, June 2006, p. 139–153.

Year Publications

Books and Monographs

[11] M. D'HONDT, L. DUCHIEN, T. MENS (editors). *Proceedings of the ERCIM Workshop on Software Evolution (EVOL'06)*, April 2006.

[12] L. DUCHIEN, C. DUMOULIN (editors). *Actes des 2ème Journées sur l'Ingénierie Dirigée par les Modèles (IDM'06)*, ISBN 10: 2-7261-1290-8, June 2006.

Doctoral dissertations and Habilitation theses

[13] R. ROUVOY. *Une démarche à granularité extrêmement fine pour la construction de canevas intergiciels hautement adaptables : application aux services de transactions*, Ph. D. Thesis, Laboratoire d'Informatique Fondamentale de Lille, Lille, France, December 2006, http://www.lifl.fr/~rouvoy/rouvoy-phd-06.pdf.

Articles in refereed journals and book chapters

[14] J. BRICHAU, R. CHITCHYAN, S. CLARKE, E. D'HONDT, A. GARCIA, M. HAUPT, W. JOOSEN, S. KATZ, J. NOYÉ, A. RASHID, M. SÜDHOLT. *A Model Curriculum for Aspect-Oriented Software Development*, in "Special Issue on Software Engineering Curriculum Development, IEEE Software", To appear, vol. 32, n^{o} 6, November/December 2006, http://www.cs.tcd.ie/Siobhan.Clarke/papers/AOCurriculum.pdf.

[15] M. A. CIBRÁN, M. D'HONDT, V. JONCKERS. *Mapping High-Level Business Rules to and Through Aspects*, in "L'Objet", vol. 12, n^{o} 2–3, April–September 2006, p. 63–88.

[16] K. GYBELS, R. WUYTS, S. DUCASSE, M. D'HONDT. *Inter-Language Reflection: A Conceptual Model and Its Implementation*, in "Computer Languages, Systems & Structures", vol. 32, n^{o} 2–3, July–October 2006, p. 109–124.

[17] R. PAWLAK. *Spoon: Compile-time Annotation Processing for Middleware*, in "IEEE Distributed Systems Online", vol. 7, n^{o} 11, November 2006.

Publications in Conferences and Workshops

[18] O. BARAIS, J. LAWALL, A.-F. LE MEUR, L. DUCHIEN. *Safe Integration of New Concerns in a Software Architecture*, in "Proceedings of the 13th International Conference on Engineering of Computer Based Systems (ECBS'06), Potsdam, Germany", IEEE, March 2006, p. 52–64.

[19] J. BRICHAU, A. KELLENS, K. GYBELS, K. MENS, R. HIRSCHFELD, T. D'HONDT. *Application-Specific Models and Pointcuts using a Logic Meta Language*, in "Proceedings of the 14th International Smalltalk Conference, Prague, Czech Republic", Lecture Notes in Computer Science, To appear, Springer, September 2006.

[20] M. A. CIBRÁN, M. D'HONDT. *A Slice of MDE with AOP: Transforming High-Level Business Rules to Aspects*, in "Proceedings of the 9th International Conference on Model Driven Engineering Languages and Systems (MoDELS'06), Genova, Italy", Lecture Notes in Computer Science, vol. 4199, Springer, October 2006, p. 170–184.

[21] M. A. CIBRÁN, M. D'HONDT. *Explicit High-Level Rules for the Customization of Web Services Management*, in "Proceedings of the NetObjectDays Conference on Objects, Aspects, Services, the Web (NODe'06), Erfurt, Germany", GI-Edition — Lecture Notes in Informatics, Bonner Köllen Verlag, September 2006, p. 113–128.

[22] C. DE ROOVER, J. BRICHAU, T. D'HONDT. *Combining Fuzzy Logic and Behavioral Similarity for Non-Strict Program Validation*, in "Proceedings of the 8th ACM- SIGPLAN International Symposium on Principles and Practice of Declarative Programming (PPDP'06), Venice, Italy", To appear, July 2006.

[23] J. DUBUS, P. MERLE. *Applying OMG D&C Specification and ECA Rules for Autonomous Distributed Component-based Systems*, in "Proceedings of the Models Workshop on Models@Runtime, Genova, Italy", To appear, October 2006.

[24] J. DUBUS, P. MERLE. *Autonomous Deployment and Reconfiguration of Component-based Applications in Open Distributed Environments*, in "Proceedings of the 8th International OTM Symposium on Distributed Objects and Applications (DOA'06), Montpellier, France", Lecture Notes in Computer Science, vol. 4277, Springer-Verlag, November 2006, p. 26–27.

[25] J. DUBUS, P. MERLE. *Vers l'auto-adaptabilité des architectures logicielles dans les environnements ouverts distribués*, in "Proceedings of the 1ère Conférence Francophone sur les Architectures Logicielles (CAL'06), Nantes, France", Hermès Sciences, September 2006, p. 13–29.

[26] J. FABRY, N. PESSEMIER. *KALA: A Domain-Specific Solution to Tangled Aspect Code*, in "Proceedings of the 1st GPCE Workshop on Domain-Specific Aspect Languages (DSAL'06), Portland, Oregon", October 2006, http://dsal06.dcc.uchile.cl/_Media/fabry.pdf.

[27] A. FLISSI, P. MERLE. *A Generic Deployment Framework for Grid Computing and Distributed Applications*, in "Proceedings of the 2nd International OTM Symposium on Grid computing, high-performAnce and Distributed Applications (GADA'06), Montpellier, France", Lecture Notes in Computer Science, vol. 4279, Springer-Verlag, November 2006, p. 1402–1411.

[28] A. KELLENS, K. MENS, J. BRICHAU, K. GYBELS. *A Model-Driven Pointcut Language for More Robust Pointcuts*, in "Proceedings of the 4th International AOSD Workshop on Software Engineering Properties of Languages for Aspect Technology (SPLAT'06), Bonn, Germany", March 2006, http://www.aosd.net/workshops/splat/2006/papers/kellens.pdf.

[29] A. KELLENS, K. MENS, J. BRICHAU, K. GYBELS. *Managing the Evolution of Aspect-Oriented Software with Model-based Pointcuts*, in "Proceedings of the 20th European Conference on Object-Oriented Programming (ECOOP'06), Nantes, France", Lecture Notes in Computer Science, vol. 4067, Springer, July 2006.

[30] F. LOIRET, D. SERVAT, L. SEINTURIER. *A First Experimentation on High-Level Tooling Support upon Fractal*, in "Proceedings of the 5th International ECOOP Workshop on Fractal Component Model (Fractal'06), Nantes, France", July 2006.

[31] T. MENS, R. VAN DER STRAETEN, M. D'HONDT. *Detecting and Resolving Model Inconsistencies Using Transformation Dependency Analysis*, in "Proceedings of the 9th International Conference on Model Driven Engineering Languages and Systems (MoDELS'06), Genova, Italy", Lecture Notes in Computer Science, vol. 4199, Springer, October 2006, p. 200–214.

[32] I. MICHIELS, C. DE ROOVER, J. BRICHAU, E. GONZALES-BOIX, T. D'HONDT. *Program Testing Using High-Level Property-Driven Models*, in "Proceedings of the 18th International Conference on Software Engineering and Knowledge Engineering (SEKE'06), San Francisco Bay, USA", To appear, July 2006.

[33] C. NOGUERA, R. PAWLAK. *AVal: an Extensible Attribute-Oriented Programming Validator for Java*, in "Proceedings of the 6th IEEE International Workshop on Source Code Analysis and Manipulation (SCAM'06), Philadelphia, PA, USA", To appear, IEEE Computer Society, September 2006.

[34] C. NOGUERA, R. PAWLAK. *Open Static Pointcuts Through Source Code Template*, in "Proceedings of the International AOSD Workshop on Open and Dynamic Aspect Languages (AOL'06), Bonn, Germany", March 2006, http://jacquard.lifl.fr/En/Pub2006?action=bibentry&bibfile=jacquard&bibref=noguera-aol-06.

[35] N. PESSEMIER, L. SEINTURIER, T. COUPAYE, L. DUCHIEN. *A Model for Developing Component-based and Aspect-oriented Systems*, in "Proceedings of the 5th International Symposium on Software Composition (SC'06), Vienna, Austria", Lecture Notes in Computer Science, vol. 4089, Springer-Verlag, March 2006, p. 259–273.

[36] N. PESSEMIER, L. SEINTURIER, T. COUPAYE, L. DUCHIEN. *A Safe Aspect-Oriented Programming Support for Component-Oriented Programming*, in "Proceedings of the 11th International ECOOP Workshop on Component-Oriented Programming (WCOP'06), Nantes, France", Technical Report, vol. 2006-11, Karlsruhe University, July 2006.

[37] R. ROUVOY, P. MERLE. *Leveraging Component-Oriented Programming with Attribute-Oriented Programming*, in "Proceedings of the 11th International ECOOP Workshop on Component-Oriented Programming (WCOP'06), Nantes, France", Technical Report, vol. 2006-11, Karlsruhe University, July 2006.

[38] R. ROUVOY, P. MERLE. *Using Microcomponents and Design Patterns to Build Evolutionary Transaction Services*, in "Proceedings of the International ERCIM Workshop on Software Evolution (EVOL'06), Lille, France", April 2006, http://www.lifl.fr/~rouvoy/uploads/English/rouvoy-ercim-06.pdf.

[39] R. ROUVOY, N. PESSEMIER, R. PAWLAK, P. MERLE. *Apports de la Programmation par Attributs au Modèle de Composants Fractal*, in "Actes des 5èmes Journées Composants (JC'06), Perpignan, France", To appear, October 2006.

[40] R. ROUVOY, N. PESSEMIER, R. PAWLAK, P. MERLE. *Using Attribute-Oriented Programming to Leverage Fractal-Based Developments*, in "Proceedings of the 5th International ECOOP Workshop on Fractal Component Model (Fractal'06), Nantes, France", July 2006.

[41] R. ROUVOY, P. SERRANO-ALVARADO, P. MERLE. *A Component-based Approach to Compose Transaction Standards*, in "Proceedings of the 5th International Symposium on Software Composition (SC'06), Vienna, Austria", Lecture Notes in Computer Science, vol. 4089, Springer-Verlag, March 2006, p. 114–130.

[42] R. ROUVOY, P. SERRANO-ALVARADO, P. MERLE. *Towards Context-Aware Transaction Services*, in "Proceedings of the 6th International Conference on Distributed Applications and Interoperable Systems (DAIS'06), Bologna, Italy", Lecture Notes in Computer Science, vol. 4025, Springer-Verlag, June 2006, p. 272–288.

[43] S. SAUDRAIS, O. BARAIS, L. DUCHIEN. *Using Model-Driven Engineering to generate QoS Monitors from a Formal Specification*, in "Proceedings of the EDOC Workshop on Advances in Quality of Service Management (AQSerM'06), Hong Kong", October 2006.

[44] L. SEINTURIER, N. PESSEMIER, L. DUCHIEN, T. COUPAYE. *A Component Model Engineered with Components and Aspects*, in "Proceedings of the 9th International SIGSOFT Symposium on Component-Based Software Engineering (CBSE'06), Västeras, Sweden", Lecture Notes in Computer Science, vol. 4063, Springer, June 2006, p. 139–153.

[45] L. SEINTURIER, N. PESSEMIER, C. ESCOFFIER, D. DONSEZ. *Towards a Reference Model for Implementing the Fractal Specifications for Java and the .NET platform*, in "Proceedings of the 5th International ECOOP Workshop on Fractal Component Model (Fractal'06), Nantes, France", July 2006.

[46] H.-M. TRAN, O. BARAIS, A.-F. LE MEUR, L. DUCHIEN. *Safe Integration of new Concerns in a Software Architecture*, in "Proceedings of the 2nd International ECOOP Workshop on Architecture-Centric Evolution (ACE'06), Nantes, France", To appear, July 2006.

[47] R. VAN DER STRAETEN, M. D'HONDT. *Model Refactorings through Rule-Based Inconsistency Resolution*, in "Proceedings of the 21st Annual ACM Symposium on Applied Computing, Model Transformation Track (SAC'06), Dijon, France", vol. 2, ACM, April 2006, p. 1210–1217.

[48] G. WAIGNIER, A.-F. LE MEUR, L. DUCHIEN. *A Generic Framework for Integrating New Functionnalities into Software Architectures*, in "Proceedings of the 2nd International ECOOP Workshop on Architecture-Centric Evolution (ACE'06), Nantes, France", July 2006.

Internal Reports

[49] FAROS. *Etat de l'art sur la contractualisation et la composition*, Technical report, n^{o} F-1.1, RNTL ANR, 1 January 2006 to 31 december 2008, aug 2006.

[50] R. PAWLAK, C. NOGUERA, N. PETITPREZ. *Spoon: Program Analysis and Transformation in Java*, Technical report, n^{o} 5901, INRIA, May 2006, http://hal.inria.fr/inria-00071366.

Miscellaneous

[51] G. DUFRÊNE. *Modèle de composants et architectures orientées services*, Technical report, Laboratoire d'Informatique Fondamentale de Lille (LIFL), Lille, France, jun 2006, http://www.lifl.fr/~dufrene/publi/dufrene-master-06.pdf.

[52] X. SUN. *Reconfiguration d'architecture pour les applications mobiles*, Technical report, Laboratoire d'Informatique Fondamentale de Lille (LIFL), Lille, France, June 2006.

[53] G. WAIGNIER. *Un cadre générique pour l'intégration de nouvelles préoccupations dans des architectures logicielles*, Technical report, Laboratoire d'Informatique Fondamentale de Lille (LIFL), Lille, France, June 2006.

References in notes

[54] J. L. PFALTZ, M. NAGL, B. BÖHLEN (editors). *Applications of Graph Transformations with Industrial Relevance, Second International Workshop, AGTIVE 2003, Charlottesville, VA, USA, September 27 - October 1, 2003, Revised Selected and Invited Papers*, Lecture Notes in Computer Science, vol. 3062, Springer, 2004.

[55] J. ARAÚJO, A. RASHID, B. TEKINERDOGAN, A. MOREIRA, P. CLEMENTS (editors). *Early Aspects 2003: Aspect-Oriented Requirements Engineering and Architecture Design*, March 2003, http://www.cs.bilkent.edu.tr/AOSD-EarlyAspects/.

[56] J. ALDRICH, C. CHAMBERS, D. NOTKIN. *ArchJava: connecting software architecture to implementation*, in "Proceedings of the 24th International Conference on Software Engineering (ICSE-02), New York", ACM Press, May 19–25 2002, p. 187–197.

[57] AS-2 EMBEDDED COMPUTING SYSTEMS COMMITTEE SAE. *Architecture Analysis & Design Language (AADL)*, November 2004, SAE Standards n o AS5506.

[58] O. BARAIS. *Construire et Maîtriser l'évolution d'une architecture logicielle à base de composants*, Ph. D. Thesis, Laboratoire d'Informatique Fondamentale de Lille, Lille, France, November 2005.

[59] O. BARAIS, E. CARIOU, L. DUCHIEN, N. PESSEMIER, L. SEINTURIER. *TranSAT: A Framework for the Specification of Software Architecture Evolution*, in "ECOOP First International Workshop on Coordination and Adaptation Techniques for Software Entities - WCAT04, Oslo - Norway", June 2004, http://wcat04.unex.es/.

[60] O. BARAIS, L. DUCHIEN. *SafArchie : Maîtriser l'Evolution d'une Architecture Logicielle*, in "Langages, Modèles et Objets - Journées Composants - LMO 2004-JC 2004, Lille - France", L'objet, vol. 10, n^{o} 2-3/2004, Hermès Sciences, March 2004, p. 103-116.

[61] O. BARAIS, L. DUCHIEN. *SafArchie Studio: An ArgoUML Extension to Build Safe Architectures*, in "Workshop on Architecture Description Languages - WADL 2004, Toulouse - France", August 2004, http://www.laas.fr/FERIA/SVF/WADL04/.

[62] A. BEUGNARD, J.-M. JÉZÉQUEL, N. PLOUZEAU, D. WATKINS. *Making Components Contract Aware*, in "Computer", vol. 32, n^{o} 7, July 1999, p. 38–45.

[63] P. BROWNE. *Using Drools in Your Enterprise Java Application*, in "OnJava", n^{o} 24-08, 2005.

[64] E. BRUNETON, T. COUPAYE, J. STEFANI. *The Fractal Component Model, version 2.0-3*, Online documentation, February 2004, http://fractal.objectweb.org/specification/.

[65] F. BUENO, D. CABEZA, M. CARRO, M. HERMENEGILDO, P. LÓPEZ-GARCÍA, G. PUEBLA. *The Ciao Prolog system. Reference manual*, Technical report, n^{o} CLIP3/97.1, School of Computer Science, Technical University of Madrid (UPM), August 1997, http://www.clip.dia.fi.upm.es/.

[66] P. COLLET, R. ROUSSEAU, T. COUPAYE, N. RIVIERRE. *A Contracting System for Hierarchical Components.*, in "CBSE", 2005, p. 187-202, http://dx.doi.org/10.1007/11424529_13.

[67] T. COUPAYE, E. BRUNETON, J.-B. STÉFANI. *The ObjectWeb Fractal Specification*, 2002, http://fractal.objectweb.org.

[68] D. DIAZ, L. SEINTURIER, L. DUCHIEN, P. FLAMENT. *Un modèle pour la séparation et la traçabilité des préoccupations*, in "Atelier Evolution du Logiciel à LMO'05, Bern, Swiss", March 2005.

[69] R. FILMAN, D. FRIEDMAN. *Aspect-Oriented Programming is Quantification and Obliviousness*, October 2000, http://citeseer.ist.psu.edu/filman00aspectoriented.html.

[70] J.-M. GEIB, C. GRANSART, P. MERLE. *CORBA : des concepts la pratique*, Masson, 1997.

[71] K. GYBELS, J. BRICHAU. *Arranging Language Features for More Robust Pattern-based Crosscuts.*, in "Proceedings of the Second International Conference on Aspect-Oriented Software Development, New York, NY, USA", ACM Press, 2003.

[72] R. HAYTON. *FlexiNet Open ORB Framework*, Technical report, APM ltd, Poseidon House, Castle Park, Cambridge, UK, 1997.

[73] INRIA. *The ObjectWeb Home Page*, 2003, http://www.objectweb.org.

[74] V. ISSARNY, C. KLOUKINAS, A. ZARRAS. *Systematic Aid for Developing Middleware Architectures*, in "Communications of the ACM", vol. 45, n^{o} 6, June 2002, p. 53 - 58.

[75] V. ISSARNY, T. SARIDAKIS, A. ZARROZ. *A Survey of Architecture Definition Languages*, Technical report, C3DS Project, June 1998.

[76] G. KICZALES, J. LAMPING, A. MENDHEKAR, C. MAEDA, C. LOPES, J. LOINGTIER, J. IRWIN. *Aspect-Oriented Programming*, in "Proceedings of the 11th European Conference on Object-Oriented Programming (ECOOP'97)", Lecture Notes in Computer Science, vol. 1241, Springer, juin 1997, p. 220-242.

[77] G. KNIESEL, T. RHO. *Generic Aspect Languages - Needs, Options and Challenges*, in "Proceedings of the 2ième Journée Francophone sur le Développement de Logiciels Par Aspects (JFDLPA 2005)", 2005.

[78] F. KON, F. COSTA, G. BLAIR, R. H. CAMPBELL. *The Case for Reflexive Middleware*, in "Communications of the ACM", vol. 45, n^{o} 6, juin 2002, p. 33 - 38.

[79] B. KOSAYBA, P. MERLE, R. MARVIE, J.-M. GEIB. *Production d'environnements graphiques à partir de méta-modèles*, in "Journée de travail du groupe OCM (GDR ALP)", Laboratoire d'Informatique Fondamentale de Lille, February 2003.

[80] LIFL, OMG. *CORBA Scripting Language Specification, version 1.0*, OMG TC Document formal/2001-06-05, Technical report, juin 2001.

[81] S. LEBLANC, R. MARVIE, P. MERLE, J.-M. GEIB. *Les intergiciels, développements récents dans CORBA, JavaRMI et les agents mobiles*, Chapter : TORBA : contrats de courtage pour CORBA ISBN: 2-7462-0432-0, Hermès Sciences, April 2002, p. 47-72.

[82] T. LEDOUX. *OpenCORBA: a Reflexive Open Broker*, in "Proceedings of the 2nd International Conference Reflexion'99, Saint-Malo, France", LECTURE NOTES IN COMPUTER SCIENCE (editor). , vol. 1616, Springer-Verlag, juillet 1999.

[83] T. LEDOUX, ET AL.. *Etat de l'art sur l'adaptabilité*, Technical report, n^{o} D1.1, Projet RNTL Arcad, December 2001.

[84] R. MARVIE. *Séparation des préoccupations et méta-modélisation pour environnements de manipulation d'architectures logicielles à base de composants*, Ph. D. Thesis, Laboratoire d'Informatique Fondamentale de Lille, December 2002.

[85] R. MARVIE, P. MERLE, J.-M. GEIB. *A Dynamic Platform for CORBA Component Based Applications*, in "First International Conference on Software Engineering Applied to Networking and Parallel/ Distributed Computing (SNPD'00)", ISBN : 0-9700776-0-2, May 2000.

[86] R. MARVIE, P. MERLE, J.-M. GEIB. *Towards a Dynamic CORBA Component Platform*, in "Proceedings of the 2nd International Symposium on Distributed Object Applications (DOA'2000), Dallas, Texas, USA", ISBN : 0-7695-0819-7, IEEE, September 2000, p. 305-314.

[87] R. MARVIE, P. MERLE, J.-M. GEIB, M. VADET. *OpenCCM : une plate-forme ouverte pour composants CORBA*, in "Actes de la 2ème Conférence Française sur les Systèmes d'Exploitation (CFSE'2)", April 2001, p. 1-12.

[88] 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, janvier 2000, 23.

[89] P. MERLE. *CorbaScript - CorbaWeb : propositions pour l'accès à des objets et services répartis*, Ph. D. Thesis, Laboratoire d'Informatique Fondamentale de Lille, Lille, 1997.

[90] M. MEZINI, K. OSTERMANN. *Conquering aspects with Caesar*, in "AOSD '03: Proceedings of the 2nd international conference on Aspect-oriented software development, New York, NY, USA", ISBN: 1-58113-660-9, ACM Press, 2003, p. 90–99.

[91] T.-A. MISSI, P. BEDU, L. DUCHIEN, H. NGUYEN, J. PERRIN. *Toward Structural and Behavioral Analysis for Component Models*, in "The FSE 2004 Workshop on Specification and Verification of Component-Based Systems - SAVCBS 2004, Newport Beach - CA - USA", November 2004, p. 138-141.

[92] OMG. *OMG Model-Driven Architecture Home Page*, http://www.omg.org/mda.

[93] OMG. *Common Object Broker Architecture Specification, Version 3.0*, OMG TC Document formal/2002-06-01, Technical report, juin 2002.

[94] R. PAWLAK. *La programmation orientée aspect interactionnelle pour la construction d'applications à préoccupations multiples*, Ph. D. Thesis, Conservatoire National des Arts et Métiers, Paris, december 2002.

[95] R. PAWLAK, J. RETAILLÉ, L. SEINTURIER. *La programmation orientée aspect pour Java/J2EE*, 2-212-11408-7, Eyrolles, 2004.

[96] R. PAWLAK, L. SEINTURIER, L. DUCHIEN, G. FLORIN, F. LEGOND-AUBRY, L. MARTELLI. *JAC : An Aspect-based Distributed Dynamic Framework*, in "Software Practice and Experience (SPE)", vol. 34, n^{o} 12, October 2004, p. 1119-1148.

[97] R. PAWLAK, L. SEINTURIER, L. DUCHIEN, L. MARTELLI, F. LEGOND-AUBRY, G. FLORIN. *Aspect-Oriented Software Development with Java Aspect Components*, in "Aspect-Oriented Software Development (AOSD)", S. CLARKE, B. FILMAN, T. ELRAD, M. AKSIT (editors). , 0-321-21976-7, Addison-Wesley, September 2004.

[98] J. D. POOLE. *Model-Driven Architecture: Vision, Standards And Emerging Technologies*, in "ECOOP 2001, Workshop on Metamodeling and Adaptive Object Models", April 2001.

[99] K. SAIKOSKI, G. COULSON, G. BLAIR. *Configurable and Reconfigurable Group Services in a Component Based Middleware Environment*, Distributed Multimedia Research Group, Department of Computing, Lancaster University, 2001.

[100] D. SCHWEISGUTH. *Second-generation aspect-oriented programming*, in "JavaWorld", July 2004, http://www.javaworld.com/javaworld/jw-07-2004/jw-0705-aop.html.

[101] D. SUVÉE, W. VANDERPERREN, V. JONCKERS. *JAsCo: an aspect-oriented approach tailored for component based software development*, in "AOSD '03: Proceedings of the 2nd international conference on Aspect-oriented software development, New York, NY, USA", ISBN: 1-58113-660-9, ACM Press, 2003, p. 21–29.

[102] G. TAENTZER. *AGG: A Graph Transformation Environment for Modeling and Validation of Software.*, in "AGTIVE", 2003, p. 446-453, http://springerlink.metapress.com/openurl.asp?genre=article&issn=0302-9743&volume=3062&spage=446.