INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team Jacquard

Weaving of Software Components

Futurs

THEME COM

Activity Report

2004

Table of contents

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).

Head of project-team
Jean-Marc Geib [Professor, USTL]

Administrative Assistant
Axelle Magnier [Assistant project INRIA since September 1st 2004]

Staff member Inria
Philippe Merle [Research associate]
Renaud Pawlak [Research associate since October 1st 2004]
Lionel Seinturier [Research associate (secondment INRIA)]

Staff member LIFL
Olivier Caron [Associate Professor Polytech'Lille]
Bernard Carré [Associate Professor Polytech'Lille]
Laurence Duchien [Professor USTL]
Anne-Françoise Le Meur [Associate Professor USTL since September 1st 2004]
Raphaël Marvie [Associate Professor USTL]
Gilles Vanwormhoudt [Associate Professor Telecom Lille I]

Ph. D. student
Olivier Barais [MESR grant]
Dolorès Diaz [NorSys CIFRE grant]
Frédéric Loiret [CEA grant since October 1st 2004]
Alexis Muller [Assistant professor, IUT, USTL]
Nicolas Pessemier [France Télécom grant since October 1st 2004]
Romain Rouvoy [INRIA-Région grant]
Mathieu Vadet [THALES CIFRE grant until December 1st 2004]

Post-doctoral fellow
Eric Cariou [Post-doctoral fellow, 2003-2004 until August 31 2004]
Patricia Serrano-Alvadarro [Post-doctoral fellow, 2004-2005 since December 1st 2004]

Project technical staff
Pierre Carpentier [Project staff - IST COACH until June 1st 2004]
Christophe Contreras [Project staff - IST COACH - ITEA OSMOSE]
Christophe Demarey [Project staff - ITEA OSMOSE until September 1st 2004]
Cédric Dumoulin [Project staff - ITEA OSMOSE since October 1st 2004]
Areski Flissi [Technical staff CNRS]
Fabien Hameau [Project staff - IST COACH until April 1st 2004]
Jérôme Moroy [Project staff - ITEA OSMOSE]
Tran-Anh Missi [Project staff - EDF until December 1st 2004]

2. Overall Objectives

Keywords: *Aspect-Oriented Programming (AOP)*, *Component Models*, *Component Weaving*, *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)*.

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 standardization organizations (OMG) and the open source software world (ObjectWeb).

2.1.1. J.M. Jacquard and the 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 will replace the fragile paper by more robust perforated cards. After that, Mr Vancanson will replace 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 machine was so precise that Joseph-Marie Jacquard designs a program that weaves his own face on a fabric. Joseph-Marie Jacquard innovations have greatly contribute to first steps of computer science with the perforated cards to support programs. The idea of independent programs for a programmatic machine was born!

3. Scientific Foundations

3.1. Weaving of Software Components

The software components challenge needs new models and platforms to allow large scale interoperability of components for designing complex distributed applications. Actually, some models exist: Enterprise Java Beans by Sun Microsystems, .Net by Microsoft and the Corba Component Model in the CORBA3 OMG standard [71]. These models and platforms are clearly not satisfactory because of the lack of functional completeness and interoperability. Moreover, the industrial propositions only deal with a lot of technical problems to capture the component software notion, but mainly forgets the needs 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 [69][73]. We agree that these points (Component Models, Component oriented Platforms and Model Driven Engineering) lead to new research problems in the goal to produce a better integrated product line from analysis to exploitation for component based applications.

Jacquard members have a great research experience in two computer science domains related with the goal of the project: Jean-Marc Geib, Philippe Merle and Raphaël Marvie have some important contributions in the Distributed Object based Platforms area [51], Laurence Duchien, Bernard Carré and Olivier Caron 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 [67] (proposed to the Scripting Language for CORBA RFP, and accepted as the CORBA Scripting Language chapter of CORBA3 [60]) 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 [72], and the View Approach for structuring the design of information systems [76].

We aim to associate 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 hardly related problems in supporting theses 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 give better software for complex applications.

For the next four years the projects 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 aspect, via a dedicated modelling tool for composition and a dynamic component and aspect oriented platform that will be a next step of our aspect 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 [53] under the name OpenCCM. Our goal is a full platform for the OMG's Corba Component Model (CCM). We want to fully capture all the aspects of this norm 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 [75] for component middleware (ObjectWeb, Inria Sardes project, France Telecom), reflexive middleware approaches (Dynamic TAO [58], Flexinet [52], OpenCorba [63], OpenORB [74]), adaptable middleware approaches (ARCAD RNTL project [62]), virtual machines (VVM) and QoS driven Midleware [54].

3.2.1. Open Middleware for the CCM

The OpenCCM project proposes an open framework to produce and exploit CORBA Components. One can specifies such a component in the new OMG IDL3 language with is an extension of the old CORBA IDL2 language. The framework can produce IDL2 schema from IDL3 descriptions, and the associated stubs for various programming languages (Java, C++, IDLscript, ...) [66]. 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. So the platform can be instantiated in several way for middleware like 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 tackled domain) from the non functional aspects (ideally programmed by an expert of the 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 (like the EJB or CCM containers) can only contain a limited set of non functional aspects (activation/termination, communications and events, security, transactions and persistency). Theses containers are not extensible neither statically nor dynamically. So they cannot respond to specific needs like fault tolerance, replication, load balancing, real- time or monitoring.

We plan to design these open containers. We investigate a generic model for containers and the weaving mechanisms which will allow an application to specify particular needs. So 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 great set 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 will 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 'on the shelves' components, a repository to manipulate and drive assemblies of components.

Current middleware propose fixed deployment strategies which are not adaptable to specific needs. These deployment tools are mainly 'black boxes' and ad-hoc in 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 the associated tools have not be realized for today in an open manner. Actually we work on a set of operations to deploy OSD assemblies. We investigate several useful properties (like optimised deployment, parallel deployment, fault tolerant deployment, transactional deployment) implemented by these operations. This will lead to an open API for adaptable deployment strategies [65]. We plan to use IDLscript to specify the strategies.

Assemblies can be constructed on demand with 'Components Off The Shelves'. We work on this point with our TORBA environment [61]. 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 its own concern (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 allow to define, at a meta level, the different concerns that we want to exist on the repository [64]. Then the environment will be able to automatic generate a new view on the repository to capture the specified activity [59]. This work will be facilitated by the works on the following topics on the project.

3.3. Aspects Oriented Design of Dynamic Components 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 and to be able to trace the evolution of this concerns in these various stages. The second one is related to dynamic features of Architecture Description Languages (ADL) [55]. The last one focuses on Aspect Oriented Programming [57] [1] in which one can capture a specific concern of a behaviour.

Finally, this project part enhances specifications of component assemblies in the goal of designing adaptable applications. We introduce integration contracts for specifying the impact of components on the application and its context. Our approach is based on AOP to specify connection and integration schemas. We also work on the JAC (Java Aspect Components) platform that provides dynamic weaving of aspects.

3.3.1. Early aspects

Business applications are faced with two main challenges. On one side they are mostly developed with an iterative process where business functionalities are added to the core application as the project requirements evolve. On the other side, 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 no integrated software process exists to take both into account. The goal of this work is thus to propose such a process and some tools to support it since the early stages of analysis and to provide features to trace their evolution them from user requirements until deployment and run-time. This work is done in the context of Dolores Diaz's PhD thesis.

3.3.2. Aspects at design-time

Software architects and designers have a reasoned frame to iteratively integrate functional and non-functional concerns into their projects, and to adapt them to unforseen functional or non-functional requirements. For helping them, analysis methods support modelling and verification tools from functional to technical architecture. Their main advantages are capacity of modelling 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 the software architecture was accepted by all the community. Various points of view on different studies bring to several approaches. These approaches focus on only one or two concerns such as component interfaces specification, behavioural analysis or software reconfiguration. So we argue that, in order to increase 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 an software architecture. It concerns three points of view: structural, functional and behavioural points of view. We use Model Driven Architecture approach with Context Independent Model and Context Specific Model. Our second goal is to introduce non functional aspects - and then connexions between components and containers - in languages for software architectures. We extend contracts between components to contracts between components and non functional components.

3.3.3. Aspects at run-time

In distributed environments, applications run in an open context. They use networks and their associated services where 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.

We have defined the Java Aspect Components (JAC) framework for building aspect-oriented distributed applications in Java [72] [9][8]. Unlike other languages like AspectJ, JAC allows dynamic weaving of aspects (aspects can be weaved or unweaved at run-time) and proposes a modular solution to specify the composition of aspects. We defined on 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 adaptable software is needed.

3.4. Functional Aspects for Components Applications

Software Engineering helps in increased productivity by re-usability. Component oriented design is a recent step towards that productivity. It allows the composition of "off the shelf" software entities, while preserving good properties on the software. The composition mechanisms are mainly used in construction and deployment phases, but the modelling phases often are not addressed by these ideas around composition.

After being considered only as documentation elements for a long time, models are gaining more and more importance in the software development lifecycle, as full software artefacts. The UML [70] standard contributes a lot to this mutation, with the identification and the structuration of models space dimensions and constructs. Models can nowadays be explicitly manipulated through metamodeling techniques, dedicated tools or processes such as the MDA [69] transformation chains. This is "Model Driven Engineering" [56].

The main motivation is the reduction of delays and costs by the capitalization of design efforts (models) at each stage, and the automation, as far as possible, of transitions between these stages. So it would be possible to separate high level business oriented models from low level architectural and technological ones, but also to reuse these models from one application to another. Indeed, once it is clear that models are full software ingredients, we are faced with new problems (needs!) such as the possibility of their reusability and composability. As a consequence, models stand more and more as good candidates for the "design for reuse" quest and specific constructs are introduced to make them generic.

We want to investigate the idea that functional decomposition of models is a way for increased re-usability. Our interest takes place in the use of functional aspects which represent the various dimensions of a tackled domain. It is related to aspect oriented structuring, and design plans like the Views, SOP [46] and Catalysis [48] approaches. We think that the scope of functional aspects can be a basis for structuring system modelling.

Our goal is to ’disconnect’ functional views from a specific domain in order to obtain functional components which will be adaptable to various contexts. This is the way to functional re-usability. Such a functional component has to capture a functional dimension with a high level of abstraction. Our idea is to introduce the notion of ’model components’ parameterized by a ’required model’ and that produce a ’provided model’. Then the modelling phase can be seen as the assembly of such components by connecting provided model to required model. Note that component ports (specified by a model) can be more sophisticated than simple interfaces of objects or software components.

As a first step, we formalized such a component model [68] and its associated design and assembly rules as an extension of the UML meta-model. We obtain adaptable model components that can be targeted to the EJB platform and the CORBA component model. We realized an implementation of this work via a UML profile. The corresponding UML Objecteering module is available at http://www.lifl.fr/~mullera.

Model parameterization is related to templates notions, such that found in the UML scope. We are exploring this notion in order to compare it to our component model. A first study shows that our model components can be expressed by UML template packages. We also identify that the UML specification needs to be extended in order to make templates parameterizable by complex models. We are defining a set of OCL constraints which formalizes this extension.

We plan to use this extension in order to define a process where package templates are composed to build a system, the way our components must do. This will lead us to define new operators for composing templates. This is related to the work of Clarke [46] around composition operations (e.g., override, merge).

A second dimension of our work is concerned with the preservation of the ’functional aspects oriented design style’ from the modelling phase to the exploitation phase. We think that the functional aspects can be transformed into software components of the underlying platform. This way gives several advantages: re-usability at the modelling phase leads to re-usability at the production phase, designers can trace the design work in the exploitation of the application. So our work can be a contribution to a seamless integration of modelling tools and component based platforms like OpenCCM or EJB. This point, preserving functional aspects into applications, was present in our earlier work on CROME [76].

We identify some structural patterns which allow to target functional decomposition onto component platforms. In [45], we present a composition-oriented approach grounded on the splitting of entities according to views requirements. Two original design patterns are formulated and capture the main issues of the approach. The first one is concerned with the management of the split component and its conceptual identity. The second offers a solution for relationships among such components. These patterns improve evolution and traceability of views and can be applied to different technological platforms.

At a practical stage, all this work is gradually integrated in Case Tools (Objecteering, Eclipse Plugin), as functional aspect oriented modelling and design facilities.

4. Application Domains

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

5. Software

5.1. Apollon

Keywords: *Graphical Editor*, *Model Driven Software Framework*, *XML*.

Participant: Christophe Contreras [correspondant].

Apollon is a model driven software framework to generate Java-based graphical editors for XML documents.

According to a XML DTD given as input, the Apollon's code generator generates a set of Java Data classes and Java Swing components implementing graphical editors for XML documents. The Java Data classes are a strongly typed reification of the XML DTD: Each XML DTD element is reified as a Java class, XML DTD children and attributes are reified as getter and setter Java methods. The Java Swing components implement the graphical representation of the Java Data classes. The graphical representation of any XML element and attribute could be customized at the generation time according to users' graphical requirements.

The Apollon's code generator is built as an extension of the open source Zeus software. The Apollon's runtime is based on the Fractal Explorer software framework described below. Apollon is already used in OpenCCM to automatically generate graphical editors for the XML DTDs defined in the OMG's CORBA Components Specification.

Apollon is a LGPL open source software available at http://forge.objectweb.org/projects/apollon.

5.2. Fractal Explorer

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

Participant: Jérôme Moroy [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 and according to end-user roles. Reactions associated to these described graphical elements could be implemented by Java classes 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 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 objects and Fractal components.

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 a LGPL open source software available at http://fractal.objectweb.org.

5.3. GoTM

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

Participant: 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, JTA, etc.). A JTA personalisation of this framework is already implemented.

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.

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

5.4. OpenCCM

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

Participant: 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 supporting various programming languages, operating systems, networks, CORBA products and vendors seamlessly. The CCM is an OMG's specification for creating distributed, server-side scalable, component-based, language-neutral, transactional, multi-users and secure applications. Moreover, one CCM application could 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 a LGPL open source software available at http://openccm.objectweb.org.

5.5. Java Aspect Components

Keywords: *Java Aspect Components*, *dynamic weaving*.

Participant: Renaud Pawlak [correspondant].

JAC (Java Aspect Components) is a project consisting in developing an aspect-oriented middleware layer. JAC current version 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 technical concerns implementations that are usually wired deep into the containers 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, profiles management, access rights checking, and authentication features. See at http://jac.objectweb.org

5.6. UML Profile

Keywords: *CCM Specification*, *UML Profile*.

Participant: Olivier Caron [correspondant].

Implementation of the UML profile for Corba Component Model.
This software enables to design a UML model corresponding to the standardized UML profile for CCM. It both checks the validity of the model and generates IDL description. This software is available as a UML Objecteering module at the following address :
http://www.lifl.fr/jacquard/software/Profil-UML-CCM/index.html.

6. New Results

6.1. Open Middleware for the CCM

6.1.1. Extensible Containers

The definition of a common component middleware which can be specialized with some technical services is a major stake in the endeavour to capitalize operational systems' functions. However, modern middleware do not provide such a specialization function, i.e. a way to build extensible containers.

In [3] we define a unified approach to build specialized component middleware by assembling software services. The analysis of the Sun Microsystem's Java 2 Enterprise Edition (J2EE) and the Object Management Group (OMG)'s CORBA Component Model (CCM) standard middleware has led us to the characterization of the specialization function. Then, we apply the software component concept to services themselves with a mind to covering the services' integration, composition and use needs. We also document a system of patterns targeted to services' architectural needs. This latter meets the quality attributes of the specialization function's architecture. This approach was implemented in CCM and we delivered a prototype in the OpenCCM platform in partnership with the European IST COACH project.

The use of software components and patterns, combined with a empirical and incremental method, rationally divides the inherent complexity of the specialization between the middleware provider, the service provider and the end-user. We note a noteworthy benefit in terms of reuse and efficiency in practice.

6.1.2. Middleware Infrastructures to Deploy Distributed Component-Based Applications

Deployment of software components for building distributed applications consists of the coordination of a set of basic tasks like uploading component binaries to the execution sites, loading them in memory, instantiating components, interconnecting their ports, setting their business and technical attributes. The automation of the deployment process then requires the presence of a software infrastructure distributed itself on the different execution sites.
[18]
presents the specification of such an infrastructure for the deployment of CORBA component-based applications. This latter is designed and implemented in the context of our OpenCCM platform, an open source implementation of the CORBA Component Model. The main characteristic lays on the fact that this infrastructure is itself designed as a set of CORBA component assemblies. This allows its dynamic assembly during its deployment over the execution sites.

Component middleware allow the automatization of applications deployment process. This function, called deployment machine, instantiates applications from their architectural descriptions. Unfortunately, currently each middleware implements its own deployment machine. Thus no capitalization is proposed as far as conceptual or implementation aspects are concerned.

To promote this capitalization, in [24] we propose a model driven approach to build component middleware deployment machines. This approach introduces a UML profile of workflow which allows us to define deployment models independently of any targeted component middleware. Such a model is then refined for each targeted middleware and the obtained model is mapped to different execution platforms. The models and transformations of this approach are illustrated on a CORBA Components deployment machine implemented using the Fractal component model.

The multiplication of architecture description languages, component models and platforms implies a serious dilemma for component based software architects. On the one hand, they have to choose a language to describe concrete configurations which will be automatically deployed on execution platforms. On the other hand, they wish to capitalize their software architectures independently of any description languages or platforms.

To solve this problem, we propose a multi personalities environment for the configuration and the deployment of component based applications. This environment is composed of a core capturing a canonical model of configuration and deployment, and a set of personalities tailored to languages and platforms. [23] details the architecture of such an environment and describes the personalities for the CORBA and Fractal component models.

6.1.3. Component-Based Software Framework for Building Transaction Services

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 increasing constantly. Each transaction service implements pieces of code that has already been written in another 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 [34] and [35], we present GoTM, a Component-Based Adaptive Middleware (CBAM) software framework. This framework enables to build various transaction services compliant with existing transaction standards. GoTM provides adaptive properties to support different transaction models and standards in the same transaction service. GoTM supports also 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.

6.1.4. Middleware Benchmarking

Nowadays, distributed Java-based applications could be built on top of a plethora of middleware technologies such as Object Request Brokers (ORB) like CORBA and Java RMI, Web Services, and component-oriented platforms like Enterprise Java Beans (EJB) or CORBA Component Model (CCM). Choosing the right middleware technology fitting with application requirements is a complex activity driven by various criteria such as economic costs (e.g. commercial or open source availability, engineer training and skills), conformance to standards, advanced proprietary features, performance, scalability, etc. Regarding performance, a lot of basic metrics could be evaluated like round-trip latency, jitter, or throughput of twoway interactions according to various parameter types and sizes.

Many projects have already evaluated these middleware performance metrics. Unfortunately, they have not compared different kinds of middleware platforms simultaneously. This could be helpful for application designers requiring to select both the kind of middleware technology to apply and the best implementation to use.

In [22], we present an experience report on the design and implementation of a simple benchmark to evaluate the round-trip latency of various Java-based middleware platforms, i.e. only measuring the response time of twoway interactions without parameters. Even if simple, this benchmark is relevant as it allows users to evaluate the minimal response mean time and the maximal number of interactions per second provided by a middleware platform. Empirical results and analysis are discussed on a large set of widely available implementations including various ORB (Java RMI, Java IDL, ORBacus, JacORB, OpenORB, and Ice), Web Services projects (Apache XML-RPC and Axis), and component-oriented platforms (JBoss, JOnAS, OpenCCM, Fractal, ProActive). This evaluation shows that our OpenCCM platform already provides better performance results than most of the other evaluated middleware platforms.

6.2. Aspect Oriented design of dynamic components assemblies

6.2.1. Aspects at design-time

First, with new component platforms, architects create distributed applications by assembling components. In all these platforms, software architecture defines the application organization as a collection of components plus a set of constraints on the interactions between components. Facing the difficulties of building correct software architecture, abstract software architecture models were built. They are powerful methods in the specification and analysis of high-level designs. Lots of architecture description models have been defined to describe, design, check, and implement software architectures. Many of these models support sophisticated analysis and reasoning or support architecture-centric development.

Nevertheless, these models are often static or work only on the components composition. In these conditions, it is difficult to build large software architecture or to integrate new components in an existing software architecture or to add a forgotten concern such as security or persistency. So, we propose SafArchie and TranSAT (Transform Software Architecture Technologies), an abstract component model for designing software architectures [36]. With SafArchie, we based our approach on architecture types that are points of reference at each step of our reasoning [36][12]. We develop SafArchie Studio [13], an architecture centric tool based on three-view perspective and driven by the component life cycle. In TranSAT [11][12], we extend the classical concepts of the software architecture models to describe technical concerns independently from a design model and integrate them step by step. These refinement steps are highly inspired by Aspect Oriented Programming (AOP), where the designer defines all the facets of an application (business and technical). To ensure a correct component composition and a correct weaving of technical components, we add behavioural information on components. These information are used to specify temporal communication protocols and check properties such as deadlock freedom or synchronization between components.

6.2.2. Architectures and points of view

ODP (Open Distributed Processing) model provides five viewpoints where each one represents one concern. They are Enterprise viewpoint (describes client needs and organisation policy), Information viewpoint (describes systems data), Computational viewpoint (describes business functionality), Engineering viewpoint (describes mechanism supporting distributed communication) and Technology viewpoint (describes technology utilisation). However, ODP is limited because it provides rich concepts for modelling distributed systems in accordance with the different viewpoints but it does not provide languages and tools for modelling and analysing software qualities and it does not guide an architect from one viewpoint to another viewpoint.

We propose a methodology for supporting modelling and analysing in the change from Computational viewpoint to Engineering viewpoint. First, we create a component model to build functional architectures in the Computational viewpoint. This component model is improved with assembly and composition concerns. Second, the Engineering viewpoint needs some concepts as hosts and channels to represent technical architectures.

Finally, one of the difficult tasks is to define non-functional properties in technical architectures. These properties such as transaction, persistence or security come from non-functional requirements. The transformation from functional to technical architecture must take into consideration these properties in order to specify the complete execution platform architecture. We have developed an architectural figure concept that represents some solutions to address these non-functional requirements [28]. These figures can be reused and help in the transformation from functional architecture to technical architecture. Moreover, we require efficient methods in order to analyze technical architecture's qualities about three analysis requirements, which are behaviour, deployment and non-functional properties [27]. Our models for constructing and analysing software architecture can apply not only in ODP but also in the architectural level of any proposed model following IEEE1471's viewpoint. Within our approach, architecture is evaluated following three principal criteria: system behaviour, deployment correctness, and non-functional qualities. The capacity of analysing software architecture with the three criteria allows for building efficient quality software.

6.2.3. Aspects and Components for Software Architectures

Software architectures need to accomodate both functional and non functional concerns. Functional concerns encompass pieces of code coming from the application domain being analyzed, and the non functional ones relate to the services provided by the environment (OS, network, application server, ...) that need to be integrated.

Component-based approaches and their associated architecture description languages are good at addressing functional software assemblies but do not provide dedicated concepts for integrating non functional needs. Conversely, aspect-oriented programming is well suited for adding some non functional crosscutting concerns into an application but does not address the issue of assembling functional parts. As both facets are almost always present in applications we then believe that there is a strong need for an approach that address both. This is the topic of the FAC action that we begun in 2004.

The goal of the FAC (Fractal Aspect Component) action [44][31][32] is to define a model for components and aspects. We envision a symmetric model in which there is only one kind of entity, component, and in which aspects are themselves components. Treating aspects as components leads to model that can be better and more easily integrated in the various stages (building, packaging, deploying, managing, debugging, ...) of the software life cycle: e.g. packaging an aspect is not different than packaging a component. The question of whether AOP should be symmetric or not (see the debate AspectJ vs Hyper/J) is open in the international research community. We believe that the asymmetric approach taken by AspectJ and others simplified the acceptance of AOP at the early stages, but that for the longer term the seamless integration of AOP requires to come back to a symmetric approach.

The second main feature of the FAC model deals with bindings. In existing ADL and component models, a binding links a client component requiring a service to a server component providing that service. Binding are most of the time dynamic in the sense that a component can be unbound and rebound to another one at runtime. This kind of binding is well suited for representing functional dependencies within an application. The FAC model supports a second kind of binding that we call crosscut binding. It links an aspect component with the components being aspectized (for instance, it links the transaction aspect with all components requiring some transactions). This binding is associated as in any other AOP tool, with a crosscut regular expression that defines the component aspectized. Hence, an application with FAC owns two kinds of binding: functional ones and crosscut ones. All these bindings can be dynamicaly introspected and modified. The advantage of such an approach is that the software architecture is made clearer and that the dependencies, both functional and crosscutting, are expressed, which is not the case with other approaches.

The FAC model is designed and developped in the context of a contract with France Telecom R&D.

6.3. Functional Aspect and MDE

6.3.1. Functional Aspects

The design of information systems remains at the present time a difficult task which requires that great number of entities and concerns be taken into account whether they are functional or not. Consequently, approaches supporting a decomposition of systems according to their functional dimensions were thus proposed at the programming level with AOP (Aspect-Oriented Programming) [57] but also at the design level with the SOD (Subject-Oriented Design) [47] or with views approaches [49].

The problem of the reuse of these functional aspects arises now. Indeed, this reuse must make it possible to improve the productivity and reliability in the field of information systems design. Various approaches propose the reuse of functional aspects in various forms, like the design of reusable frameworks [48] or in the form of UML templates [46].

In [29] we compare techniques for composing and parameterizing models and keep the advantages of the later ones to specify reusable functional aspects.

Parameterization capabilities are offered by the UML Template notion. Applications are numerous and various, with the result that its initial introduction in UML1.3 was deeply revisited and strengthened in the UML2 standard. Though its specification remains much more structural and verbal in [70]. Particularly, constraints lack for the precise definition of the related "binding” relation which allows to obtain models from templates. These constraints are needed to verify the correctness of the resulting models. That is why we propose in [20] a set of OCL constraints [77] which could strengthen the notion of model templates and facilitate the above verification. An Eclipse plugin was realized. This new plugin enables to build new metamodels with OCL constraints and then generate Java implementations of these metamodels which both support the creation and verification of models based on these OCL constraints.

At a design level, we have defined an independent platform model of view-components which enables to describe complex information systems involving numerous functional dimensions. To target this model on specific platforms, we propose an approach which is based on several design patterns. We start from our patterns supporting views through split representation of entities [45]. A first experimentation in the Fractal component model is presented in [14] which allows to envision the usage of Fractal controllers to manage split components. In [21] we focus on the reuse of functional aspects or views at the implementation level using adaptation techniques. The reuse of views is ensured by applying the adapter pattern [50]. We show how to compose the views pattern and the adaptation one. The result provides an implementation of reusable functional aspects that can be composed at the exploitation stage.

6.3.2. Model Driven Engineering

In the context of model driven engineering (MDE), model are the corner stone of software engineering processes. Then, model have to be properly defined in order to be useful and to be the basis of software production. In order to assist the designer in defining a model as well as to control the result, we have studied the chaining and composition of model transformations.

A modeling process is made of several steps (for example identifying components, then their relations, and finally the services used and provided by each component). Delta-trans framework has been defined to support such well defined modeling processes. Defining a model is then seen as a set of transformations (like creating model elements, defining relations between existing model elements) applied on an empty model. Based upon the definition of a modeling process, tools are automatically produced in order to support the process. The modeling tool is dedicated to a process [42].

In order to easily support software processes in a particular domain (like telecoms or health care) product lines are factorizing the common concerns of application while supporting their adaptation to particular needs using configurable concerns. The goal of our study regarding the composition of model transformations is to support the building of software production lines. Transformations are primitive or composite (composition of transformations). Transformations are provided as executable components, then easily composable to define MDE-based product lines. Our experimental framework *picotin* provides modeling means for defining and composing model transformations. A set of tools rely on these definitions to generate transformation component implementations. Finally, a prototype environment support the execution of these transformations [39].

Both studies are complementary in that the first one provides means for modeling a system respecting domain constraints, and the second provides means for actually building the system from its definition. This approach is currently studied together with Alicante (a company working in the field of health care) in order to build tools for GPs based upon their basic data items (person, medical act, temperature, etc) and specific constraints ($36C < temperature < 42C$).

7. Contracts and Grants with Industry

7.1. RNTL ACCORD

The RNTL project ACCORD has 8 participants (EDF, CNAM, ENST, ENST-Bretagne, France Télécom, Inria, LIFL, Softeam). The goal is to produce a design framework using explicit contracts to specify and assemble components. We want to facilitate understanding of complex systems, enhance the flexibility and the reusability of components, and allow strong validation of assemblies. This work has to be done independently of technical infrastructures.

7.2. France Telecom

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. Nicolas Pessemier PhD thesis work is directly related to this contract.

7.3. NorSys

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 the 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 company to tell whether it can also bring to innovations into the early stages of requirement engineering, analyze and design. This contract begun in January 2004. Dolores Diaz PhD thesis work is directly related to this contract.

8. Other Grants and Activities

8.1. Regional Initiatives

8.1.1. IRCICA

The ’Region Nord Pas de Calais’ has initiated a large research plan around the new technologies for communications. We lead the software section of this plan. Beyond this plan the ’Region Nord Pas de Calais’ has facilitated the creation of a new research institut called IRCICA to promote new collaborative research projects between software and hardware laboratories. The Jacquard project is one of the first projects supported by this institut.

8.1.2. MOSAIQUES

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 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 headed of this project

8.2. National Initiatives

8.2.1. AS MDA

The specific action MDA, created in June 2003 et funded by CNRS, studies the interest of the Model Driven Architecture approach. The standard promoted by OMG is only one example. The aim of this AS is to organize the research community in this domain in order to understand and to help the industrial community in an approach that it can be a significant evolution on the middle and long term. This AS includes IRIN, LIFL, LSR, IMAG, I3S, PRIM and CEA laboratories.

8.3. European Initiatives

8.3.1. ObjectWeb

ObjectWeb is an European initiative 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. We are member of this consortium, and Fractal Explorer, GoTM, JAC, and OpenCCM are projects hosted by the consortium.

8.3.2. IST COACH

The ’COmponent based ArCHitecture for distributed telecom applications’ project is a PCRDT project in the IST program. The project groups 9 academic and industrial labs. The goal is a component oriented CORBA platform for the telecom domain. Our OpenCCM platform forms the basis of this architecture.

8.3.3. ITEA OSMOSE

OSMOSE stands for ’Open Source Middleware for Open Systems in Europe’. It is an ITEA project. The project groups 16 European industrials and 7 public labs. The goal is to give up an European dimension for the ObjectWeb consortium. The OSMOSE project wants to federate high quality components from European labs, and to produce applications for the great European industrial domains.

8.3.4. AOSD-Europe

AOSD-Europe is an ongoing proposal to set up a Network of Excellence (NoE) on aspect-oriented software development within IST-FP6. The proposal brings together 11 research groups and among them members of the Jacquard project and other members from OBASCO, Pop-Art and Triskell Inria projects. The proposal 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.

8.4. International Initiative

8.4.1. OMG

We work in the international consortium OMG (Object Management Group) 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 CCM (CORBA Component Model) chapter for which we lead the response group and the revision task force. We also participate in the definition of a UML profile for CORBA Components.

Philippe Merle is:

- Chair of the OMG Components 1.2 Revision Task Force (RTF).
- Member of the OMG Deployment Revision Task Force (RTF).
- Member of the OMG UML Profile for CCM Finalization Task Force (FTF).
- Member of submission team for the OMG UML Profile for CCM RFP.
- Member of the voting list for the OMG MOF 2.0 IDL RFP.
- Member of the voting list for the OMG MOF 2.0 Query/View/Transf. RFP.
- Member of the voting list for the OMG MOF 2.0 Versioning RFP.
- Member of the voting list for the OMG QoS for CORBA Components RFP.
- Member of the voting list for the OMG Streams for CCM RFP.

8.4.2. AOP Alliance

AOP Alliance is an international open-source initiative to provide a common API for building aspect weavers <http://aopalliance.sourceforge.net>. This initiative has been launched and is led by Jacquard's members (R. Pawlak and L. Seinturier). The goal is to bring together several aspect framework creators, to analyse the functional requirements that are shared by all these frameworks, and to set up a common API. This API allows to modularise the writing of aspect weavers, and promotes the reuse of common building blocks between AOP frameworks. AOP Alliance brings together leaders of international recognized AOP projects such as JAC, Spring, PROSE. The API is in beta-stage but is already implemented in JAC and PROSE.

9. Dissemination

9.1. Scientific community animation

9.1.1. Examination Commitees

- **Jean-Marc Geib** was in the examination committee of the following PhD thesis :
  - M. Vadet, November 2004, University of Lille (adviser)
  - E. Renaux, December 2004, University of Lille (adviser)
  - D. Touzet, March 2004, University of Rennes (referee)
  - C. Nebut, November 2004, University of Rennes (referee)
  - H. Zheng, November 2004, University of Lille (chair)
  - M. Figeac, December 2004, University of Lille (chair)
  - K. Drira (HDR), December 2004, University of Toulouse (referee)
- **Laurence Duchien** was in the examination committee of the following PhD thesis :
  - A.-T. Le, January 2004, University of Grenoble (referee)
  - K. Macedo, March 2004, University of Rennes (referee)
  - D. Fauthoux, June 2004, University of Toulouse (referee)
  - L. Quintian, July 2004, University of Nice (referee)
  - R. Lenglet, November 2004, University of Grenoble (referee)
  - O. Nano, December 2004, University of Nice (referee)
- **Philippe Merle** was in the examination committee of the following PhD thesis :
  - M. Vadet, November 2004, University of Lille (co-adviser)
  - H Cervantes, March 2004, University of Grenoble (referee)
  - A. Ribes, December 2004, University of Rennes (referee)
- **Olivier Caron** was in the examination committee of the following PhD thesis :
  - E. Renaux, December 2004, University of Lille (co-adviser)

9.1.2. Journals, Conferences, Workshop

- **Jean-Marc Geib** has been a member of the following programm committees:
  - DECOR 2004, 1st Francophone conference on software Deployment and (Re)configuration, October 2004, Grenoble
  - NOTERE 2004, Les Nouvelles technologies de la répartition, June 2004, Sadia, Maroc
- **Laurence Duchien** has been a member of the following programm committees:
  - JC 2004, Journées composants, March 2004, Lille
  - JFPLA 2004, Journées Francophones sur le développement de logiciels par aspects, September 2004, Paris
  - Workshop ECOOP Abstract Communications on Distributed Systems, Oslo, June 2004
  - ADVICE Journal
- **Philippe Merle** has been a member of the following programm committees:
  - DECOR 2004, 1st Francophone conference on software Deployment and (Re)configuration, October 2004, Grenoble
  - LMO 2004, Langages Models and Objects conference, March 2004, Lille
  - JC 2004, Journées Composants, March 2004, Lille
  - IJCA, special issue, June 2004
  - JMAC 2004, Journées Multi-Agents et Composants, November 2004, Paris
- **Bernard Carré** has been a member of the following programm committees:
  - LMO 2004, Langages Models and Objects conference, March 2004, Lille
- **Lionel Seinturier** has been a member of the following programm committees:
  - JC 2004, Journées Composants, March 2004, Lille
  - JFPLA 2004, Journées Francophones sur le développement de logiciels par aspect, September 2004, Paris
  - DECOR 2004, 1st Francophone conference on software Deployment and (Re)configuration, October 2004, Grenoble
  - ADVICE Journal
- **Renaud Pawlak** has been a member of the following programm committees:
  - workshop AOSD ACP4IS, AOD Conference March 2004, Lancaster, UK.
  - ADVICE Journal

9.1.3. Miscellaneous

- The Jacquard Team has organized the LMO Conference and The JC Workshop in Lille (15-18 March 2004).
- Bernard Carré and Jérome Euzenat (INRIA Rhône-Alpes) are co-editors of the Special volume of the journal 'L'Objet' (Vol 10/2-3, 2004).
- Renaud Pawlak gives an invited seminar on Recombining programming at Demeter research unity of Northeastern University, Boston, US.

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 teaches 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 is in charge of the Master Professionnel Sciences et Technologies Mention Informatique - M2 - Spécialité IAGL at USTL, UFR IEEA.

Raphaël Marvie Object Oriented Design, Distributed System Design and C++ programming in Master GMI, Distributed System Design in Master MIAGE-FC, Advanced Technologies for Distribution in Master Pro IPI NT at USTL, UFR IEEA.

Anne-Françoise Le Meur teaches Databases and the Internet, Design of distributed application, and C programming at USTL, UFR IEEA.

Bernard Carré teaches OO design and programming at Polytech'Lille (USTL Engineer school). He is in charge of the Computer Sciences and Statistics Department of Polytech'Lille.

At Polytech'Lille Engeneering school, **Olivier Caron** is in charge of the following modules: Data Bases and Distributed Software Components. He also teaches Object-Oriented Programming and Operating Systems.

Gilles Vanwormhoudt teaches Algorithms and Programming in C, 1th year of the engineering program, Design and Programming of Distributed Applications, 3rd year of the engineering program and Technologies for Web development, 5th year of the engineering program, ENIC Telecom Lille 1. He is in charge of Multimedia computing and engineering specialization, 5th year of the engineering program, and the Project-Conferences-Report module for the "Multimedia computing and engineering" specialization, ENIC Telecom Lille 1.

Jacquard team's members participate to some Research Masters of Computer Science (University of Lille, University of Paris 6, University of Montpellier, University of Valenciennes and RPI University, Hartford, US) on the CCM, MDE and on AOP.

9.3. Miscellaneous

Jean-Marc Geib is the head of the LIFL laboratory CNRS 8022, the head of the CIM axis, member of the UFR board at USTL, member of CSE 27nd section of Universities of Lille 1, Lille 2 and Littoral. He is member of the management commitee TACT (Technologies Avancées pour la Communication et les Transports) of the Etat-Region Contract Nord-Pas-de-Calais and the coordinator of the communication program since 2004. He is IRCICA cofounder (Intitut de Recherche sur les Composants matériels et logiciels pour l'information et la communication avancée) that is a research federation between LIFL (Science computing), l'IEMN (electronics) and the PhLAM (photonique). He is in charge of the RTP Distributed Systems of the CNRS STIC Dept.

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 scientific commitee of the national ACI Security.

10. Bibliography

Books and Monographs

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

Doctoral dissertations and Habilitation theses

[2] E. RENAUX. *Définition d'une démarche de conception de systèmes à base de composants*, Ph. D. Thesis, Laboratoire d'Informatique Fondamentale de Lille I, Lille, December 2004.

[3] M. VADET. *Un Modèle de Services Logiciels pour la Spécialisation des Intergiciels à Composants*, Ph. D. Thesis, Laboratoire d'Informatique Fondamentale de Lille - Thalès, Lille, France, December 2004.

Articles in referred journals and book chapters

[4] A. BEUGNARD, E. CARIOU. *Composants de Communication et Middleware*, Vuibert, September 2004.

[5] O. CARON, E. RENAUX, J.-M. GEIB. *Vers de Véritables Composant EJB Réutilisables*, in "Coopération dans les systèmes à objets - Numéro spécial de la revue l'Objet", vol. 10, n^o 1, 2004, p. 73-87.

[6] R. MARVIE. *Langages de description d'architectures: un état de l'art*, to appear, Hermès Sciences, 2004.

[7] R. MARVIE. *Vers des patrons de méta-modélisation*, in "Interopérabilité des systèmes distribués: des modèles aux intergiciels, Special Number of Technique et Science Informatique - TSI", to appear, 2004.

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

[9] 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.

[10] R. PAWLAK, H. YOUNESSI. *On Getting Use Cases and Aspects to Work Together*, in "Journal of Object Technology (JOT)", http://www.jot.fm/issues/issue200401, vol. 3, n^o 1, February 2004, 26.

Publications in Conferences and Workshops

[11] 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.

[12] 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.

[13] 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.

[14] O. BARAIS, A. MULLER, N. PESSEMIER. *Extension de Fractal pour le Support des Vues au sein d'une Architecture Logicielle*, in "Objets Composants et Modèles dans l'ingénierie des SI - OCM-SI 2004, Biarritz - France", June 2004.

[15] X. BLANC, O. CARON, A. GEORGIN, A. MULLER. *Transformation de modèles : d'un modèle abstrait aux modèles CCM et EJB*, in "Langages, Modèles et Objets - LMO 2004, Lille - France", L'objet, vol. 10, n^{o} 2-3/2004, Hermès Sciences, March 2004, p. 161-174.

[16] S. BONNET, R. MARVIE, J.-M. GEIB. *Putting Concern-Oriented Modeling into Practice*, in "2nd Nordic Workshop on UML, Modeling, Methods and Tools - MWUML 2004, Turku - Finland", August 2004.

[17] S. BONNET, O. POTONNIÉE, R. MARVIE, J.-M. GEIB. *A Model-Driven Approach for Smart Card Configuration*, in "Proceedings of the Third International Conférence on Generative Programming and Component Engineering - GPCE 2004, Vancouver - Canada", Lecture Notes in Computer Science, Springer-Verlag, October 2004.

[18] F. BRICLET, C. CONTRERAS, P. MERLE. *Une infrastructure à composants pour le déploiement d'applications à base de composants CORBA*, in "1ère conférence sur le Déploiement et la (Re)Configuration de Logiciels - DECOR 2004, Grenoble - France", Hermès Sciences, October 2004.

[19] E. CARIOU, R. MARVIE, L. SEINTURIER, L. DUCHIEN. *OCL for the Specification of Model Transformation Contracts*, in "OCL and Model-Driven Engineering UML 2004 Workshop", October 2004.

[20] O. CARON, B. CARRÉ, A. MULLER, G. VANWORMHOUDT. *Formulation of UML 2 Template Binding in OCL*, in "7th International Conference on UML Modeling Groups and Applications - UML 2004, Lisbon - Portugal", October 2004.

[21] O. CARON, B. CARRÉ, A. MULLER, G. VANWORMHOUDT. *Mise en oeuvre d'aspects fonctionnels réutilisables par adaptation*, in "Première journée Francophone sur le Développement de Logiciels par Aspects - JFDLPA 2004, Paris - France", September 2004.

[22] C. DEMAREY, G. HARBONNIER, R. ROUVOY, P. MERLE. *Benchmarking the Round-Trip Latency of Various Java-Based Middleware Platforms*, in "The OOPSLA 2004 Component And Middleware Performance Workshop - CMP 2004, Vancouver - Canada", Studia Informatica, October 2004.

[23] A. FLISSI, P. MERLE. *Vers un environnement multi-personnalités pour la configuration et le déploiement d'applciations à base de composants logiciels*, in "1ère conférence sur le Déploiement et la (Re)Configuration de Logiciels - DECOR 2004, Grenoble - France", Hermès Sciences, October 2004.

[24] A. FLISSI, P. MERLE. *Une démarche dirigée par les modèles pour construire les machines de déploiement des intergiciels à composants*, in "Langages et Modèles à Objets - LMO 2005, Berne - Suisse", to appear, Hermès Sciences, March 2005.

[25] A. GEORGIN, F. LEGOND-AUBRY, S. MATOUGUI, N. MOTEAU, A. MULLER, A. TAUVERON, J. THIBAUT, B. TRAVERSON. *Description des assemblages et des contrats pour la conception par composants*, in "Journées Composants, Lille - France", March 2004.

[26] B. KOSAYBA, R. MARVIE, J.-M. GEIB. *Model Driven Production of Domain-Specific Modeling Tools*, in "4th OOPSLA Workshop on Domain-Specific Modeling, Vancouver - Canada", October 2004.

[27] 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.

[28] T.-A. MISSI, L. DUCHIEN, P. BEDU, H. NGUYEN, J. PERRIN. *Figures de transformation pour Architectures Logicielles*, in "Langages, Modèles et Objets - LMO 2005, Bern - Switzerland", L'objet, to appear, vol. 11, n^o 2-3/2005, Hermès Sciences, March 2005.

[29] A. MULLER. *Reusing functional aspects : from composition to parameterization*, in "Aspect-Oriented Modeling Workshop - AOM 2004, Lisbon - Portugal", October 2004.

[30] T. MURALI, R. PAWLAK, H. YOUNESSI. *Applying Aspect Orientation to J2EE Business Tiers Patterns*, March 2004.

[31] N. PESSEMIER, L. SEINTURIER, L. DUCHIEN, O. BARAIS. *Une extension de Fractal pour l'AOP*, in "Première journée Francophone sur le Développement de Logiciels par Aspects - JFDLPA 2004, Paris - France", September 2004.

[32] N. PESSEMIER, L. SEINTURIER, L. DUCHIEN. *Components ADL and AOP: Towards a Common Approach*, in "Workshop ECOOP Reflection; AOP and Meta-Data for Software Evolution - RAM-SE 2004, Oslo - Norway", June 2004.

[33] E. RENAUX, O. CARON, J.-M. GEIB. *Chaîne de production de systèmes à base de composants logiques*, in "Langages, Modèles et Objets - LMO 2004, Lille - France", L'objet, vol. 10, n^o 2-3/2004, Hermès Sciences, March 2004, p. 147-160.

[34] R. ROUVOY, P. MERLE. *GoTM : Vers un canevas transactionnel à base de composants*, in "Langages, Modèles et Objets - LMO 2004, Lille - France", L'objet, vol. 10, n^o 1-3/2004, Hermès Sciences, March 2004, p. 131-146.

[35] R. ROUVOY, P. MERLE. *Towards a Model Driven Approach to build Component-Based Adaptable Middleware*, in "The 3rd Workshop on Reflective and Adaptive Middleware - RAM 2004, Toronto - Canada", ACM, October 2004.

Internal Reports

[36] O. BARAIS, L. DUCHIEN, L. SEINTURIER. *SafArchie ADL : Construire et Déployer une Architecture Logicielle Typée*, Research Report LIFL, n^o 2004-n10, Laboratoire d'Informatique Fondamentale de Lille, September 2004.

[37] E. CARIOU, R. MARVIE, L. SEINTURIER, L. DUCHIEN. *Model Transformation Contracts and their Specification in UML and OCL*, Technical report, n° 2004-8, LIFL, April 2004.

[38] R. MARVIE, L. DUCHIEN, M. BLAY-FORNARINO. *Vers une caractérisation de la notion de plate-forme d'exécution*, Technical report, Annexe du rapport de synthèse de l'Action Spécifique CNRS sur MDA, October 2004.

[39] R. MARVIE. *A Transformation Composition Framework for Model Driven Engineering*, Technical report, n° 2004-10, LIFL, 2004.

[40] R. MARVIE. *MDA, Model Transformations and Platforms: Advocating Technological Jumps*, Technical report, n° 2004-09, LIFL, 2004.

[41] R. PAWLAK, C. CUESTA, H. YOUNESSI. *Recombinant Programming*, Technical report, n° 5380, INRIA Futurs - Projet JACQUARD, November 2004.

Miscellaneous

[42] G. BLONDEEL. *Etude et définition d'un cadre logiciel pour l'enchaînement de transformation de modèles*, Technical report, Laboratoire d'Informatique Fondamentale de Lille, June 2004.

[43] F. BRICLET. *Canevas Logiciel Générique pour le déploiement dans les intergiciels pour composants*, Technical report, Laboratoire d'Informatique Fondamentale de Lille, Lille, France, June 2004.

[44] N. PESSEMIER. *Une approche unifiée pour des Architectures Logicielles avec des besoins Fonctionnels et non Fonctionnels*, Technical report, Laboratoire d'Informatique Fondamentale de Lille, Lille, France, June 2004.

Bibliography in notes

[45] O. CARON, B. CARRÉ, A. MULLER, G. VANWORMHOUDT. *A Framework for Supporting Views in Component Oriented Information Systems*, in "International Conference on Object-Oriented Information Systems, Geneva - Switzerland", Lecture Notes in Computer Sciences, vol. 2817, Springer Verlag, September 2003, p. 164-178.

[46] S. CLARKE. *Extending standard UML with model composition semantics*, in "Science of Computer Programming, Elsevier Science", 2002.

[47] S. CLARKE, W. HARRISON, H. OSSHER, P. TARR. *Subject-Oriented Design: Towards Improved Alignment of Requirements, Design and Code*, in "Objecteed-Oriented Programming Systems, Languages and Applications (OOPSLA), Denver", November 1999.

[48] D. D'SOUZA, A. WILLS. *Objects, Components and Frameworks With UML: The Catalysis Approach*, Addison-Wesley, 1999.

[49] L. DEBRAUWER. *Des vues aux contextes pour la structuration fonctionnelle de bases de données à objets en CROME*, Ph. D. Thesis, Laboratoire d'Informatique Fondamentale de Lille I, Lille, décembre 1998.

[50] E. GAMMA, R. HELM, R. JOHNSON, J. VLISSIDES, G. BOOCH. *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Westley Professional Computing, USA, 1995.

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

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

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

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

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

[56] S. KENT. *Model Driven Engineering*, in "Proceedings of IFM 2002", Lecture Notes in Computer Science, vol. 2335, Springer-Verlag, May 2002, p. 286-298, http://www.cs.kent.ac.uk/pubs/2002/1594.

[57] 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, June 1997, p. 220-242.

[58] F. KON, F. COSTA, G. BLAIR, R. H. CAMPBELL. *The Case for Reflexive Middleware*, in "Communications of the ACM", vol. 45, n° 6, June 2002, p. 33 - 38.

[59] 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.

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

[61] 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.

[62] T. LEDOUX, ALL. *Etat de l'art sur l'adaptabilité*, Technical report, n° D1.1, Projet RNTL Arcad, December 2001.

[63] 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, July 1999.

[64] 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.

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

[66] 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.

[67] 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.

[68] A. MULLER, O. CARON, B. CARRÉ, G. VANWORMHOUDT. *Réutilisation d'aspects fonctionnels : des vues aux composants*, in "Proceedings of LMO 03", Hermès Sciences, January 2003, p. 241-255.

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

[70] OMG. *OMG UML Home Page*, http://www.uml.org.

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

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

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

[74] 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.

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

[76] G. VANWORMHOUDT. *CROME : un cadre de programmation par objets structurés en contextes*, Ph. D. Thesis, Laboratoire d'Informatique Fondamentale de Lille I, Lille, 1999.

[77] J. WARMER, A. KLEPPE. *The Object Constraint Language – Second Edition, Getting Your Models Ready for MDA*, Addison-Wesley, 2003.