INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team JACQUARD

Weaving of Software Components

Futurs

THEME 1A

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]

Staff member Inria
Philippe Merle [Research scientist]
Lionel Seinturier [Research scientist (secondment INRIA)]

Staff member LIFL
Olivier Caron [Teaching Assistant Polytech'Lille]
Bernard Carré [Teaching Assistant Polytech'Lille]
Laurence Duchien [Professor USTL]
Raphaël Marvie [Teaching Assistant USTL]
Gilles Vanwormhoudt [Teaching Assistant Telecom Lille I]

Ph. D. student
Olivier Barais [MESR]
Sylvain Leblanc [RNTL]
Alexis Muller [RNTL]
Romain Rouvoy [INRIA-Région]
Mathieu Vadet [CIFRE THALES]

Post-doctoral fellow
Eric Cariou [Post-doctoral fellow, 2003-2004]
Renaud Pawlak [Post-doctoral fellow, 2003-2004]

Project technical staff
Christophe Contreras [Project staff - IST COACH]
Christophe Demarey [Project staff - ITEA OSMOSE]
Areski Flissi [Technical staff CNRS]
Fabien Hameau [Project staff - IST COACH]
Jérôme Moroy [Project staff - ITEA OSMOSE]

2. Overall Objectives

Key words: *component models*, *separation of concerns*, *aspect oriented programming*, *component weaving*, *model driven engineering*, *run-time containers*, *integrated tools for production and exploitation of software components*.

The JACQUARD project tackles the large problem of designing complex distributed applications, i.e. composed of numerous cooperative and distributed software components which are constraint by various requirements like persistency, security, fault tolerance. We want to illustrate the capability of software engineering to produce new component oriented platforms and new methodological and technical traits to design and exploit these applications. We explore the use of component models, separation of concerns and weaving in the different phases (model, design, assembly, deployment, and execution) of the application's life cycle. We goal to produce fully functional platforms and tools, in relation with standardization organisations like OMG, and with the open source software world.

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 [61]. 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 give 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 century XIX. The machine was so precise that J.M. Jacquard designs a program that weaves his own face on a fabric. J.M. 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 in designing complex distributed applications. Actually, some models exist : Enterprise Java Beans by Sun Microsystems [26], .Net by Microsoft [66] and the Corba Component Model [70] 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 [55][62]. We agree that theses 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 [30][52], 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 [53] (proposed to the Scripting Language for CORBA RFP, and accepted as the IDLscript chapter of CORBA3 [69] [45]) 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 [27][28], [4], and the View Approach for structuring the design of information systems [67].

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:

- 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.

- Produce a methodological and technical environment to specify theses assemblies of components using aspect oriented design, via a dedicated modelling tools for assemblies and a dynamic aspects oriented platform (in 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 [68] 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 [72] for component middleware (ObjectWeb, INRIA Sardes project, France Telecom), reflexive middleware approaches (Dynamic TAO [64][37], Flexinet [32], OpenCorba [40], OpenORB [21][65]), adaptable middleware approaches (ARCAD RNTL project [63]), virtual machines (VVM) and QoS driven Midleware [33].

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, ...) [51]. 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 onto 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 of 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, 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 some problems: 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 [50][49]. We plan to use IDLscript to specify the strategies.

Assemblies can be construct on demand with 'Components On The Shelves'. We work on this point with our TORBA environment [46][48][47][39][38], [9]. 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 preoccupation (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 preoccupations that we want to exist on the repository [43][41][44][42], [15]. Then the environment will be able to automatic generate a new view on the repository to capture the specified activity[14]. 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 following the context. Changes can occur in an asynchronous manner and the behaviour has to be adapted without human actions and without application stops. 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 these two points : new specifications of assemblies to allow the description of dynamically evolving applications, and new platforms to allow dynamic evolution at run-time. The first point is related to Architecture Description Languages (ADL) [34], but these languages often are only focused on static aspects. The second point is related to Aspect Oriented Programming [36] in which one can capture a specific aspect of a behaviour, but AOP platforms are mainly restricted to static weaving and do not allow dynamic manipulation of aspects.

This part of the project wants to enhance specifications of assemblies of components in the goal of designing adaptable applications. We introduce 'integration contracts' to specify the impact of the components on the application and its context. Our approach is based on AOP to specify connexion and integration schemas. We also work on the JAC (Java Aspect Component) platform to put in work the dynamic weaving of aspects. This is described in the next sections.

3.3.1. Dynamic Software Architectures

Our first goal is to propose enhancements of a component models to allow specifications of dynamic evolution of an software architecture. It concerns three points of view: structural, functional and behavioural points of view. We follow a MDA approach with Context Independent Model and Context Specific Model. We work on some notions like 'composite' [19][20] or 'group' [31][54][65][35] [27] as good levels to specify dynamic interactions between components. Our second goal is to introduce non functional aspects - and then connexions between components and containers - in languages for software architectures. We plan to extend contracts between components to contracts between components and non functional components [29].

3.3.2. Dynamic Weaving

JAC (Java Aspect Components) is a framework in Java for programming aspects [4][5], [59][56][60][57][58]. Unlike other languages like AspectJ, JAC allows dynamic weaving of aspects (aspects can be weaved or un-weaved at run-time) and proposes a modular solution to specify the composition of aspects. Jac is distributed

under the LGPL licence at http://jac.objectweb.org. This version includes aspects for persistency, remote communications, GUI and logging.

We plan to extend the current version to new basic aspects for distributed programming. We plan also to integrate JAC in an UML based CASE tool to promote Aspect Oriented Design. In a second step we plan to evolve JAC towards Component Oriented Programming. JAC will produce component and containers (like in the EJB, CCM, Web Services or Fractal models).

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 'on the shelves' 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 touched by these ideas around composition. The MDA approach is a new way to touch the modelling phase with innovative structuring ideas. It promotes the design of computer systems around models and iterative transformations of these models towards the final application. It specifies the use of Platform Independent Models and Platform Specific Models. And finally the MDA approach claims for an automatic process, based on transformations, to product applications from models. This is clearly an innovative way to challenge the design of software engineering tools, but this is an immature process which needs some work to understand the key concepts under this approach.

We want to investigate the idea that model oriented approaches are also a way for increased re-usability. It is related to aspect oriented structuring, and design plans like the Views, SOP [24] and Catalysis [25] approaches. Our interest takes place in the use of functional aspects which represent the various dimensions of a tackled domain. 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, and , by this way, 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 that simple interfaces of objects or software components.

In a first step we consider UML like models. Then our approach requires the study of assembly constraints at the model level. We plan to use OCL based constraints. Connections between models will often need some adaptations of the models. We plan to use transformation tools to realize these adaptations. This is related to the work of Clarke around composition operations (override, merge, ...). We are exploring the UML template notion in order to compare it to our model components. Our present work shows that the UML specification needs to be extended in order to make templates parameterizable by complex models. We are defining a set of constraints which formalizes this extension.

We also plan to use MOF mechanisms or UML profile mechanisms to capture the modelling of our components.

Associated with the weaving techniques of AOD, we investigate the possibility of preserving the 'functional aspects oriented design' 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 [67][22][23].

We identify some structural patterns which allow to target functional decomposition onto component platforms.

Finally we plan to introduce the 'functional aspect oriented modelling' in Case Tools (UML Objecteering and Eclipse UML2 plugin). A first result is the specification and realization of a UML profile which allows to

describe view components. These view components can be connected to different base models according to 'design and connections' rules. An Objecteering module is available at http://www.lifl.fr/~mullera.

5. Software

5.1. OpenCCM

Participant: Philippe Merle [correspondant].

Key words: *Corba Component Model*, *OpenCCM platform*.

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. Morover, one CCM application could be deployed and run on several distributed nodes simultaneously.

Then OpenCCM allows you to design, implement, compile, package, assemble, deploy, install, instantiate, configure, execute, and manage distributed CORBA component-based applications.

See at http://openccm.objectweb.org.

5.2. Java Aspect Components

Participant: Laurence Duchien [correspondant].

Key words: *Java Aspect Components*, *dynamic weaving*.

JAC (Java Aspect Components) is a project consisting in developing an aspect-oriented middleware layer. JAC current version is 0.11. 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.

6. New Results

6.1. OpenCCM

>From the beginning of the project, a lot of efforts has been put on the design and implementation of OpenCCM, our Open CORBA Components Platform. Currently, OpenCCM is composed of five independent and complementary parts: The production tool chain, the packaging and assembly tool chain, the distributed deployment infrastructure, the container runtime framework, and the management framework.

The OpenCCM production tool chain allows users to design, define, implement and compile CORBA components. This tool chain is based on a component-based modular and extensible architecture [2].

The heart of this architecture is the OpenCCM Interface Repository which is the runtime reification of the CORBA Components meta model. This repository stores meta data reifying component types and all other OMG IDL/PSDL/CIDL constructions. This meta data repository provides application programming interfaces (API) conform with the OMG specification of the CORBA 3.0 Interface Repository. Then a set of front-end compilers and back-end generators are connected to this repository to feed and visit it respectively.

On one hand, the OpenCCM repository is fed by several front-end compilers for the OMG Interface Definition Language 3.0 (OMG IDL) allowing the design of component types, for the OMG Persistent State Definition Language (OMG PSDL) allowing the design of persistent states, and for the OMG Component Implementation Definition Language (OMG CIDL) allowing the design of component implementations. Moreover, OpenCCM contains also a front-end compiler for the OMG UML Profile for CORBA Components allowing users to design component types and implementations from a UML tool. This latest front-end is built following the OMG Model Driven Architecture (MDA) approach. A UML repository is built from the UML meta model thanks to the ModFact tool (a OMG MOF repository factory) developed by the LIP6 (http://modfact.lip6.fr). Our team has contributed to ModFact in order to correct and improve it to support MOF features required in OpenCCM. The generated UML repository is fed with XMI UML files produced from any UML tool. Then the mapping from UML concepts to OpenCCM concepts is viewed as a model transformation which visits the UML repository to feed the OpenCCM Interface Repository. This transformation is reversible to allow users to generate XMI UML files from OpenCCM meta data.

On the other hand, the OpenCCM repository is visited by a set of back-end generators for producing the CORBA Components client and server equivalent OMG IDL 2.x mapping, OMG PSDL related Java interfaces and implementations, OMG CIDL related Java container code, Java templates for implementing CORBA components, and finally pretty-printers for OMG IDL 3.0, OMG PSDL, OMG CIDL, and XMI UML. Back-end generators for XML descriptors defined by the CORBA Components Specification are planned for next year. Most of these generators are designed as MDA model transformations, especially those generating Java code. Here a Java repository has been built to reifying Java interfaces and classes to generate. Then various specific transformations visit the OpenCCM Interface Repository and feed the Java repository according to the Java code to generate. Finally a general transformation visits the Java repository in order to generate Java files.

One main advantage of the OpenCCM production tool chain architecture is its modularity and extensibility allowing users to add new front-end compilers for other component design notations (e.g. Architecture Description Languages, Platform Independent Models, Domain Specific Languages, etc), and to add new back-ends to generate new output files like documentation, C++ code, etc. Currently, this modular and extensible OpenCCM production tool chain architecture has been validated by external users. For instance, the OpenCCM tool chain is embedded in the PERCO platform developed by THALES Communications, and in the open source Cadena tool developed by the Kansas University (http://cadena.projects.cis.ksu.edu/). This latest is a plug-in for the Eclipse IDE allowing graphical design of real-time and embedded CORBA component-based distributed applications. Here Cadena feeds the OpenCCM Interface Repository according to graphical design of components, and then Cadena invokes the various OpenCCM back-ends to generate OMG IDL/CIDL component descriptions and Java based component implementations automatically.

The OpenCCM packaging and assembling tool chain allows users to package and assemble CORBA components. From the users' point of view, this is a graphical stand-one packaging and assembling tool allowing the edition of CORBA component and assembly ZIP archives and all CORBA Components XML descriptors. Internally, this tool is built with our Appolon software framework (http://forge.objectweb.org/projects/appolon) composed of a set of code generators and a runtime. Appolon is a generative approach producing Java Data Classes and Java Swing components according to any XML DTD provided as input. The generated 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 generated Java Swing components implement the graphical representation of the generated Java Data classes. To allow customisation of the produced code, Appolon generators are parametizable via a XML document allowing users to customize the graphical representation of any XML element and attribute. Then in the context of OpenCCM, Appolon is used to generate most of the graphical part of the packaging and assembling tool according to the four XML DTD defined by the CORBA Components Specification. Here this generative approach allows us to add and/or experiment extensions to the CCM XML DTD easily without breaking the packaging and assembling tool each time. Moreover this allows end-users to add their own XML extensions and to produce

their own customized packaging and assembling tool specific to their requirements. At runtime, the generated Java components are executed on top of our generic browser software framework described later.

The OpenCCM distributed deployment infrastructure allows users to deploy, install, instantiate, configure, and interconnect CORBA components in order to build CORBA component-based distributed applications. This infrastructure takes CORBA component and assembly ZIP archives as input and interprets their XML descriptors in order to set up the CORBA distributed applications, i.e. download component binaries on target nodes, create required CORBA components servers and containers, install CORBA component homes into containers, create CORBA component instances from CORBA component homes, configure CORBA component properties, interconnect CORBA components via their ports, register homes and components into the CORBA Naming Service and the CORBA Trading Service, and finally start the created CORBA components.

The OpenCCM distributed deployment infrastructure is built as a CORBA component-based distributed application which implements the deployment API defined by the CORBA Components Specification and provides new supervision and management features. The main CORBA components of this deployment infrastructure are the DCIManager reifying a deployment domain (e.g. a local network), the NodeManager reifying a virtual node where CORBA components could be downloaded and instantiated, the ComponentServerManager reifying a CORBA component server where containers could be created, the ContainerManager reifying a CORBA component container where homes could be installed and components created, the AssemblyModelManager reifying an installed CORBA component assembly, and the AssemblyManager reifying a running CORBA component assembly. The main goal of this latest component is to implement the distributed deployment process: It schedules distributed deployment operations according to XML CORBA Component Assembly descriptors. As the OpenCCM deployment infrastructure is a set of CORBA components, it is possible to create various deployment infrastructure architectures by defining various assemblies of the previous described components. Moreover it is possible to deploy a basic deployment infrastructure and then use it to deploy more sophisticated deployment infrastructures which will meet application specific requirements.

Finally as the possible OpenCCM deployment infrastructures are composed of CORBA components running into CORBA containers, then this is possible to inject non functional properties like persistency, transactions, security, etc. into the distributed deployment process. Currently, we have experimented transactional distributed deployment which allows us to automatically rollback deployments when failures are detected, e.g. deployment errors, network and node crashes, etc. Our future perspectives are to experiment the injection, into the distributed deployment process, of some new non functional properties like access control security to control that deployments are only done by authorized persons, and various deployment scheduling policies to inject flexibility and parallelism into the distributed deployment process.

The OpenCCM container runtime framework allows to host and execute CORBA components, and inject non functional properties thanks to containers. This runtime framework is built on top of the CORBA 2.4 features, i.e. Object Request Broker (ORB), Portable Object Adapter (POA), and Portable Interceptors (PI), and on top of standard CosNaming, CosTrading, CosTransactions, and CosPersistentState services.

The runtime library currently implements session containers and it is composed of a set of Java classes inherited by Java container code generated by the OpenCCM production tool chain. Moreover, a flexible and extensible container model and architecture has been designed to allow us to build new containers supporting new services as logging. The container services are designed as software components exposing clear external interfaces, they can be packaged and deployed. Then a new container is dynamically built at deployment time by assemblying several container service components.

Related to the persistency service, we have implemented the CORBA Persistent State Service (PSS) on top of the Java Data Objects (JDO) technology. This allows us to run the persistency service on top of various JDO implementations, currently the Kodo commercial product (http://www.solarmetric.com/) is supported and the ObjectWeb Speedo open source implementation (http://speedo.objectweb.org) should be supported soon.

Related to the transaction service, we have initiated a new research collaborative activity inside the ObjectWeb community. This new activity, named GoTM (http://forge.objectweb.org/projects/gotm), targets to design a component-based software framework to build transactional functionalities like transaction monitors

and resource managers [18][8][7]. This software framework is designed on top of the ObjectWeb Fractal component model and is implemented on top of the ObjectWeb Julia reference implementation. A first prototype design and implementation of this GoTM framework has been publicly published to the ObjectWeb community.

The OpenCCM management framework allows users to discover, introspect, manage, monitor and reconfigure CORBA components dynamically at runtime. >From the users' point of view, this is a graphical stand-one management tool allowing the exploration and management of the OpenCCM Interface Repository, of the CosNaming and CosTrading services, of all the components of the OpenCCM distributed deployment infrastructure, and of any application specific CORBA object, home, and component. Internally, this tool is built on top of a generic software framework for building graphical user interface management browsers. This framework is extensible by plug-ins. A plug-in is described by a XML file and contains a set of Java code implementing API defined by the browser software framework. The XML files allow users to configure how to navigate on and display objects and to set up which menu items are associated to objects. This browser software framework has been experimented in several contexts: The OpenCCM browser, the OpenCCM packaging and assembling tool, the Fractal browser and the GoTM browser. In the context of OpenCCM, this framework allows us to provide generic management capabilities, and users could add their own application specific management features.

Finally, all the previously presented OpenCCM features are supported on a large set of environments: Java 1.2.1, 1.3.x and 1.4.x environments ; Linux, Linux for PDA, MacOS X, Solaris, Windows 2000/NT/XP and CE for PDA operating systems ; and Borland Enterprise Server (BES) 5.0.2 and 5.2, IONA ORBacus 4.1.x, JacORB 2.0, and the Community OpenORB 1.2.1, 1.3.0, 1.3.1 and 1.4.0 CORBA implementations.

6.2. Aspects

Since the beginning of the project, two new research directions have been followed and lead to the following set of results.

Many systems in safety critical areas use replication of vital components to improve reliability. Replication have been studied extensively in the literature. So, we do not invent new replication techniques but we deal with the software development complexity introduced by such fault tolerance form. We investigate new ways for handling replication in distributed applications using separation of concerns. Actually, SoC is achieved by modularising the software around its logical functionalities or sub-functionalities. Using modularisation allows the programmer to decompose the software into understandable components and achieve a better degree of reusability and maintainability. Our goal is to design a flexible software infrastructure able to implement replication as independent as possible of the context of its application. To do that, we use the Aspect Oriented Programming approach which aims at allowing different aspects of complex software to be programmed separately and then woven into a single final application.

A library of aspect components have been defined for distributed programming with JAC. They allow to program aspect-oriented applications with availability needs. They provide an active replication policy with broadcast, load-balancing, cache and consistency aspects. Our approach decomposes first the replication management application into many communication components defined separately. We provide then a same generic programming interface to integrate the components into the application as *aspects*. We allow thus an application programmer to choose the adequate manner to implement its replication strategy (replication type, replication degree, replica location, communication and synchronization forms...) , without having to know the implementation details of the replication management protocol [6].

Second, 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 TranSAT (Transform Software Architecture Technologies), an abstract component model for designing software architectures[10]. In TranSAT, 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.

6.3. Functional Aspects

Software engineering aims at being rationalized always more and begin to reach levels of productivity and reuse that come near to other fields such as mechanics or electronics. Objects technologies then those based on components participate to this quest. In [3], we focus upon the design of reusable components that are adaptable in their functional or business dimension (aspect). We start from views oriented design approaches to make these views reusable in different contexts (information systems). The resulting component model and its associated rules of design and assembly are formalized in an extension of the UML meta-model. We obtain adaptable components of models. That can be targeted to the EJB platform and the CORBA component model. We realized an implementation of this work via an UML profile. The corresponding UML Objecteering module is available at http://www.lifl.fr/~mullera

The Component Oriented Design of Information Systems is spreading. After being used for gaining in reusability at the architectural level, components are nowadays applied at the business logic level. We focus on the design of multiple functional views in such information systems, specially within the EJB framework. Traditionally, in the database context, this problem is solved by the notions of ”view-schemas” applied to a database schema. In [1], 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. Finally, we apply these patterns to the EJB framework. This framework improves evolution and traceability of views.

In the context of the RNTL ACCORD project [12], we developed some tools [16] which will be some important parts of a future chain of model production: from an abstract model with functional aspects to specific platform models such as EJB and CCM models. We have specified an UML profile for CORBA Components [17]. We also defined a methodological process to transform PIM models into PSM models by using annotating models [11].

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. RNTL IMPACT

The partners of this project are INRIA, BULL, France Télécom, Schlumberger, Scalagent, ExperLog, Kelua, I3S, LAMIH, LIFL, LIP6, LSR. The project wants to specify and construct an adaptable framework for the design of middleware. This framework includes the following components: the Fractal model of component, JORM for persistency, JOTM for transactions, JONATHAN for the bindings, JORAM for the event based communications. The project contributes to the ObjectWeb code base.

7.3. ACI GRID RMI

The collaborative action GRID RMI includes the following INRIA projects: Paris, ReMap, Jacquard, and OASIS. The goal is to tackle the software system layer of the GRID computing using objects and components techniques. The project uses different platforms: PM2, PadicoTM and PaCO++, OpenCCM, ProActive and Do!.

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.

8.2. National Initiatives

8.2.1. ObjectWeb

ObjectWeb is a French 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, we lead the College of Architects (2002-2003), and OpenCCM and JAC are projects hosted by the consortium.

We participated to the organization and animation of four ObjectWeb Architecture Meetings:

- INRIA Rhône-Alpes, Montbonnot Saint-Martin, France, 30 - 31 January 2003.
- University of Jussieu, Paris, France, 28 - 29 April 2003.
- LIP6, Paris, France, 1 - 2 July 2003.
- INRIA Rhône-Alpes, Montbonnot Saint-Martin, France, 24 - 26 September 2003.

8.3. European Initiatives

8.3.1. 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.

8.3.2. 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.

8.3.3. 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 INRIA. 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 IDLscript chapter of CORBA3) 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 an UML profile for CORBA Components.

9. Dissemination

9.1. Leadership within scientific community

9.1.1. 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 IDLscript chapter of CORBA3) 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 an UML profile for CORBA Components.

- Philippe Merle, *" Chartering the Components 1.2 Revision Task Force "*, OMG Technical Meeting, Paris, France, June 2003.
- Philippe Merle, Chair of the OMG Components 1.2 Revision Task Force (RTF),
- Philippe Merle, Member of the OMG Deployment Finalization Task Force (FTF),
- Philippe Merle, Member of the OMG UML Profile for CCM Finalization Task Force (FTF),
- Philippe Merle, Member of submission team for the OMG UML Profile for CCM RFP
- Philippe Merle, Member of the voting list for the OMG MOF 2.0 IDL RFP,
- Philippe Merle, Member of the voting list for the OMG MOF 2.0 Query/View/Transf. RFP,
- Philippe Merle, Member of the voting list for the OMG MOF 2.0 Versioning RFP,
- Philippe Merle, Member of the voting list for the OMG QoS for CORBA Components RFP,
- Philippe Merle, Member of the voting list for the OMG Streams for CCM RFP,

9.1.2. 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. Discussions have begun to move the AOPAlliance initiative to ObjectWeb.

9.2. Teaching

We give some invited tutorials in conference:

- Marc Born (Fraunhofer FOKUS), Tom Ritter (Fraunhofer FOKUS), and Philippe Merle, *" Realization of Distributed Applications using MDA and the CORBA Component Model "*, Tutorial at the ACM/IFIP/USENIX International Middleware Conference (Middleware 2003), Rio de Janeiro, Brazil, 16 - 20 June 2003. http://middleware2003.inf.puc-rio.br
- R. Pawlak, L. Seinturier, *" JAC Tutorial "*, Tutorial at AOSD'2003 (Aspect Oriented Software Development), Boston, USA.

Philippe Merle strongly participates to the 4th Summer School on Middleware and Building Distributed Applications, organised by INRIA Rhône-Alpes and ObjectWeb Consortium (ICAR 2003, 25 - 29 August 2003, Autrans, France). See at http://sardes.inrialpes.fr/ecole/2003.

We participate to some Masters of Computer Science (University of Lille, University of Valenciennes, University of Namur (Belgium), on the CCM, MDA and on AOP.

10. Bibliography

Major publications by the team in recent years

[1] 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 », 2003.

[2] S. LEBLANC, P. MERLE. *Towards Middleware Product-Lines.* in « The First Workshop on Model-driven Approaches to Middleware Applications Development (MAMAD 2003), Proceedings of the ACM/IFIP/USENIX International Middleware Conference (Middleware 2003) », pages 268 - 272, Rio de Janeiro, Brazil, June, 2003, ISBN 85-87926-03-9.

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

[4] R. PAWLAK, L. SEINTURIER, L. DUCHIEN, G. FLORIN. *JAC: A Flexible and Efficient Solution for Aspect-Oriented Programming in Java.* in « Reflection 2001, LNCS 2194 », pages 1-24, september, 2001.

[5] R. PAWLAK, L. SEINTURIER, L. DUCHIEN, G. FLORIN. *JAC : un framework pour la programmation orientée aspect en Java.* in « L'Objet, Hermès », number 4, volume 8, 2002, pages 72-92.

[6] R. PAWLAK, L. SEINTURIER, L. DUCHIEN, L. MARTELLI, F. LEGOND-AUBRY, G. FLORIN. *Aspect-Oriented Software Development with Java Aspect Components.* Addison-Wesley, 2003.

[7] M. PROCHAZKA, R. ROUVOY, T. COUPAYE. *On Enhancing Component-Based Middleware with Transactions.* in « Proceedings of On The Move to Meaningful Internet Systems 2003: OTM 2003 Workshops », Springer-Verlag LNCS, pages 1-2, Catania, Sicile, October, 2003, ISBN 3-540-20494-6.

[8] R. ROUVOY, P. MERLE. *Abstraction of Transaction Demarcation in Component-Oriented Platforms.* in « Proceedings of the fourth ACM/IFIP/USENIX International Middleware Conference (Middleware 03) », Laboratoire d'Informatique Fondamentale de Lille, Laboratoire d'Informatique Fondamentale de Lille, pages 305-323, June, 2003, ISBN 3-540-40317-5.

Articles in referred journals and book chapters

[9] S. LEBLANC, P. MERLE, J.-M. GEIB. *Les contrats de courtage TORBA.* in « L'Informatique Professionnelle », number 212, March, 2003, pages 40-44, ISSN 0750-1080.

Publications in Conferences and Workshops

[10] O. BARAIS, L. DUCHIEN, R. PAWLAK. *Separation of Concerns in Software Modeling: A Framework for Software Architecture Transformation.* in « IASTED International Conference on Software Engineering Applications (SEA) », november, 2003.

[11] X. BLANC, O. CARON, A. GEORGIN, A. MULLER. *Transformation de modèles : d'un modèle abstrait aux modèles CCM et EJB.* in « Proceedings of LMO'04 », March, 2004, toappear.

[12] 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 ceonception par composants.* in « Journées Composants », march, 2004, toappear.

[13] B. KOSAYBA, R. MARVIE, P. MERLE, J.-M. GEIB. *Production of Domain Oriented Graphic Modeling Environments.* in « Workshop on Model Driven Architecture: Foundations and Applications (MDAFA 2003) », University of Twente, Enschede, Netherlands, June, 2003.

[14] 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.

[15] R. MARVIE. *Vers des pattrons de méta-modélisation.* in « Journée de travail du groupe OCM (GDR ALP) », February, 2003.

Internal Reports

[16] O. CARON, A. MULLER. *Spécification du profil UML d'assemblage cible CCM.* Technical report, Projet RNTL Accord, june, 2003.

[17] OMG. *UML Profile for CORBA Components.* Object Management Group, May, 2003, http://www.omg.org/cgi-bin/doc?mars/03-05-09, OMG Document mars/03-05-09.

Miscellaneous

[18] R. ROUVOY. *Canevas pour le transactionnel dans les plates-formes à composants.* Technical report, DEA, Laboratoire d'Informatique Fondamentale de Lille, June, 2003.

Bibliography in notes

[19] O. BARAIS. *Approche statique, dynamique et globale de l'architecture d'applications réparties.* Technical report, DEA, Laboratoire d'Informatique Fondamentale de Lille, july, 2002.

[20] O. BARAIS, L. DUCHIEN. *OPAD: Outils pour Architectures Dynamiques.* in « Journées Composants Adaptables », october, 2002.

[21] G. S. BLAIR, G. COULSON, P. ROBIN, M. PAPATHOMAS. *An Architecture for Next Generation Middleware.* in « Proceedings of the IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing (Middleware'98) », SPRINGER-VERLAG, editor, 1998.

[22] O. CARON, B. CARRÉ, L. DEBRAUWER. *Contextualization of OODB Schemas in CROME.* in « DEXA 2000, 11th International Conference, London », september, 2000.

[23] O. CARON, B. CARRÉ, L. DEBRAUWER. *CromeJava: une implémentation du modèle CROME de conception par contextes pour les bases de données à objets en Java.* in « Proceedings of LMO'00 », janvier, 2000.

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

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

[26] L. DEMICHIEL, L. YALINALP, S. KRISHNAN. *Enterprise Java Beans Specification Version 2.0 - Public Draft.* Sun Microsystems, Mai, 2000.

[27] L. DUCHIEN. *Modèles de Programmation, Services Systèmes et Réflexivité pour la Coopération de Groupes d'Objets Répartis.* Ph. D. Thesis, Grenoble, december, 1999.

[28] L. DUCHIEN, L. SEINTURIER. *Evolutions des plates-formes orientées objets répartis.* in « Réseaux et Systèmes Répartis, Calculateurs Parallèles », L. Duchien and L. Seinturier, H. HERMÈS, editor, 2000.

[29] P. GAHIDE, N. BOURAQADI, L. DUCHIEN. *Promoting Component Reuse by Integrating Aspects and Contracts in an Architecture Model.* First AOSD Workshop on Aspects, april, 2002.

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

[31] W. HARRISON, H. OSSHER. *Member-Group Relationships Among Objects.* in « FOAL 2002 Proceedings: Foundations of Aspect-Oriented Languages Workshop at AOSD 2002 », series Technic Report, number 02-06, Department of Computer Science, Iowa State University, G. T. LEAVENS, R. CYTRON, editors, pages 9–16, avril, 2002, ftp://ftp.cs.iastate.edu/pub/techreports/TR02-06/TR.pdf.

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

[33] V. ISSARNY, C. KLOUKINAS, A. ZARRAS. *Systematic Aid for Developing Middleware Architectures.* in « Communications of the ACM », number 6, volume 45, juin, 2002, pages 53 - 58.

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

[35] *JavaGroups - A Reliable Multicast Communication Toolkit for Java.* 2001, http://www.cs.cornell.edu/Info/Projects/JavaGroupsNew/.

[36] 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) », series Lecture Notes in Computer Science, volume 1241, Springer, pages 220-242, juin, 1997.

[37] F. KON, F. COSTA, G. BLAIR, R. H. CAMPBELL. *The Case for Reflexive Middleware.* in « Communications of the ACM », number 6, volume 45, juin, 2002, pages 33 - 38.

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

[39] S. LEBLANC, P. MERLE. *TORBA : vers des composants de courtage.* in « LMO 2002 », pages 185-201, Janvier, 2002.

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

[41] R. MARVIE, J. GEIB, P. MERLE. *Separation of Concerns in Modeling Distributed Component Based Architectures.* in « Proceedings of the Sixth IEEE International Enterprise Distributed Object Computing Conference (EDOC 2002) », pages 144-154, September, 2002, ISBN: 0-7695-1742-0.

[42] R. MARVIE, L. KOZAKOV, Y. DOGANATA. *Towards Adaptive Knowledge Middleware.* in « Proceedings of the International Conference on Information and Knowledge Engineering (IKE'02) », June, 2002.

[43] R. MARVIE. *CODeX : proposition pour la description dynamique d'architectures à base de composants logiciels.* in « Actes des Journées composants: flexibilité du système au langage », pages 79-88, October, 2001.

[44] 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.

[45] R. MARVIE, P. MERLE, J.-M. GEIB, C. GRANSART. *CCM + IDL script.* in « Evolution des plates-formes orientées objets répartis, numéro spécial de Calculateurs Parallèles et », number 1, volume 12, 2000, pages 75-104, Ed. Hermès, ISBN: 2-7462-0169-0.

[46] R. MARVIE, P. MERLE, J. GEIB, S. LEBLANC. *TORBA : Trading Contracts for CORBA.* in « Proceedings of the 6th USENIX Conference on Object- Oriented Technologies and Systems (COOTS'01) », ACM/IFIP/USENIX, pages 1-14, January, 2001, ISBN : 1-880446-12-X.

[47] R. MARVIE, P. MERLE, J.-M. GEIB, S. LEBLANC. *TORBA : vers des contrats de courtage.* in « Electronic Journal on Network and Distributed Processing », number 11, volume 1, March, 2001, pages 1-18, ISSN: 1262-3261.

[48] R. MARVIE, P. MERLE, J.-M. GEIB, S. LEBLANC. *Type-safe Trading Proxies Using TORBA.* in « Proceedings of the 5th International Conference on Autonomous Distributed Systems (ISADS'01) », IEEE, pages 303-310, Dallas, Texas, USA, March, 2001, ISBN : 0-7695-1065-5.

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

[50] 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) », May, 2000, ISBN : 0-9700776-0-2.

[51] 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) », pages 1-12, April, 2001.

[52] R. MARVIE, M. PELLEGRINI. *Modèles de composants, un état de l'art.* in « Coopération dans les systèmes à objets, Numéro spécial de la revue l'Objet », number 3, volume 8, September, 2002, pages 61-90, ISBN: 2-7462-0520-3.

[53] 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.

[54] A. MONTRESOR. *Jgroup Tutorial and Programmer's Manual.* Technical report, University of Bolona, octobre, 2000.

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

[56] R. PAWLAK, L. DUCHIEN, G. FLORIN, L. MARTELLI, L. SEINTURIER. *Distributed Separation of Concerns with Aspects Components.* in « TOOLS Europe 2000, IEEE Computer Society », pages 276-287, june, 2000.

[57] R. PAWLAK, L. DUCHIEN, G. FLORIN, L. MARTELLI, L. SEINTURIER. *Une approche pour la programmation répartie : les composants d'aspect.* in « L'Objet, Hermès », number 3, volume 8, 2002, pages 39-59.

[58] 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.

[59] R. PAWLAK, L. SEINTURIER, L. DUCHIEN, G. FLORIN, L. MARTELLI. *Towards a Language for Groups of Distributed Objects.* in « Reflexive Middleware 2000 (RM2000) », april, 2000.

[60] R. PAWLAK, L. SEINTURIER, L. DUCHIEN, G. FLORIN. *Dynamic Wrappers : Handling the Composition Issue with JAC.* in « Proceedings TOOLS-USA , IEEE », pages 56-65, july, 2001.

[61] PERRAULT. *Pour une histoire de l'ordinateur.* http://www.uqtr.uquebec.ca/˜perrault.

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

[63] M. RIVEILL. http://www.essi.fr/˜riveill/ARCAD/, 2001.

[64] M. ROMAN, F. KON, R. CAMPBELL. *Design and Implementation of Runtime Reflection in Communication Middleware: The Dynamic TAO Case.* in « Proceedings of the International Conference on Distributed Computing Systems (ICDCS'99) », mai, 1999.

[65] 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.

[66] T. THAI, H. LAM. *.NET Framework Essentials.* O'Reilly, 2001.

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

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

[69] LIFL, OMG. *CORBA Scripting Language Specification, version 1.0.* Object Management Group, Boston, USA, juin, 2001, OMG TC Document formal/2001-06-05.

[70] OMG. *CORBA Components Specification, Version 3.0.* Object Management Group, juin, 2002, OMG TC Document formal/2002-06-65.

[71] OMG. *Common Object Broker Architecture Specification, Version 3.0.* Object Management Group, juin, 2002, OMG TC Document formal/2002-06-01.

[72] THIERRY COUTAGE AND ERIC BRUNETON AND JEAN-BERNARD STÉFANI. *The ObjectWeb Fractal Specification.* 2002, http://www.objectweb.org/fractal.