INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

# *Project-Team Triskell*

# *Model Driven Engineering for Component Based Software*

*Rennes*

THEME COM

Activity Report

2006

# Table of contents

# 1. Team

**Scientific head**
Jean-Marc Jézéquel [ Professor Université de Rennes 1, HdR ]

**Administrative assistant**
Loïc Lesage [ TR Inria ]

**Inria staff**
Benoit Baudry [ Research scientist Inria ]
Didier Vojtisek [ Research engineer Inria ]

**Faculty member Université de Rennes 1**
Noël Plouzeau [ Assistant Professor Université de Rennes 1 ]
Olivier Barais [ Assistant Professor Université de Rennes 1 since September 2006 ]

**Visiting Scientist**
Pierre-Alain Muller [ Assistant Professor Université de Mulhouse ]
Yves Le Traon [ France telecom R&D, HdR ]
Cristina Vicente [ Universidad de Murcia, Spain ]
Diego Caceres [ Universidad de Murcia, Spain ]
Greg O'Keefe [ Australia National University ]

**Post-doctoral Fellow**
Olivier Barais [ Assistant Professor Université de Rennes 1 until September 2006 ]

**Technical staff**
Zoé Drey [ Inria Associated Engineer until september 2006 ]
Franck Chauvel [ Université Rennes 1 (project FAROS) from October 2006 ]
Cyril Faucher [ Inria (project Openembedd) from June 2006 ]
Vincent Mahé [ Inria (project Openembedd) from October 2006 ]
François Tanguy [ Inria Associated Engineer from October 2006 ]
David Touzet [ Inria (project Systematic) ]
Marc Skipper [ Inria (project Speeds) from July 2006 ]

**Lecturer**
Damien Pollet [ until March 2006 ]

**PhD Students**
Erwan Brottier [ CIFRE grant ]
Romain Delamarre [ Brittanny Council grant since October 2006 ]
Franck Fleurey [ MENRT grant ]
Marouane Himdi [ CIFRE grant ]
Jacques Klein [ INRIA grant ]
Martin Monperus [ DGA grant ]
Jean-Marie Mottu [ MENRT grant ]
Sébastien Saudrais [ INRIA grant ]
Jim Steel [ INRIA grant ]

**Interns Students**
Sagar Sen [ University of Montreal, Quebec ]
Rodrigo Ramos [ University of Pernambuco, Brazil ]
Waqas Ahmed Saeed [ KTH, Sweden ]
Pankaj Bhatia [ RWTH Aachen, Belgium ]

# 2. Overall Objectives

## 2.1. Overall Objectives

**Keywords:** *Components*, *MDA*, *UML*, *aspects*, *contracts*, *design patterns*, *frameworks*, *objects*, *requirements engineering*, *scenarios*, *software product lines*, *test*, *validation*.

### *2.1.1. Research fields*

In its broad acceptation, Software Engineering consists in proposing practical solutions, founded on scientific knowledge, in order to produce and maintain software with constraints on costs, quality and deadlines. In this field, it is admitted that the complexity of a software increases exponentially with its size. However on the one hand, the size itself of the software is on average multiplied by ten every ten years, and on the other hand, economic pressures pushed towards reducing the duration of developments, and in increasing the rates of modifications made to the software.

To face these problems, today's mainstream approaches build on the concept of component based software. The assembly of these components makes it possible to build families of products (a.k.a. *product lines*) made of many common parts, while remaining opened to new evolutions. As component based systems grow more complex and mission-critical, there is an increased need to be able to represent and reason on such assemblies of components. This is usually done by building models representing various aspects of a product line, such as the functional variations, the structural aspects (object paradigm), of the dynamic aspects (languages of scenarios), without neglecting of course non-functional aspects like quality of service (performance, reliability, etc.) described in the form of contracts. Model Driven Engineering (MDE) is then a sub-domain of software engineering focusing on reinforcing design, validation and test methodologies based on the automatic processing of multi-dimensional models.

### *2.1.2. Project-team Presentation Overview*

The research domain of the Triskell project is the reliable and efficient design of software product lines using Model Driven Engineering. Triskell is particularly interested in component based, embedded systems with quality of service constraints.

Triskell's main objective is to develop model-based methods and tools to help the software designer to efficiently obtain a certain degree of confidence in the reliability of component assemblies that may include third-party components. This involves, in particular, investigating modeling languages allowing specification of both functional and non-functional aspects for software engineering activities ranging from requirements to detailled design. It also involves building a continuum of tools which make use of these models, from model validation and verification, automatic application of design patterns, to test environments and on-line monitors supervising the behavior of the components in a distributed application. Since these modeling languages and associated tools appear quite open-ended and very domain specific, there is a growing need for *"tools for building tools for building software"*. Triskell is hence developping KerMeta as an original meta-meta modeling approach allowing the user to fully define his modeling languages (including dynamic semantics) and associated environments (including interpreters, compilers, importers/exporters, etc.) within Eclipse.

To avoid the pitfall of developping *"tools for building tools for the sake of it"*, the Triskell project also has the goal of explicitly connecting its research results to industrial problems through collaborations with industry and technology transfer actions. This implies, in particular, taking into account the industrial standards of the field, namely the Eclipse Modeling Framework EMF, the OMG's Meta-Object Facility MOF and Unified Modeling Language UML, as well as domain specific component models.

Triskell is at the frontier of two fields of software: the field of specification and formal proof, and that of design which, though informal, is organized around best practices (*e.g.; separation of concerns with aspects, models, design patterns, or the use of off-the-shelf components*). We believe that the use of our techniques will make it possible to improve the transition between these two worlds, and will contribute to the fluidity of the processes of design, implementation and testing of software.

# 3. Scientific Foundations

## 3.1. Overview

The Triskell project studies new techniques for the reliable construction of software product lines, especially for distributed and reactive software. The key problems are components modeling and the development of formal manipulation tools to refine the design, code generation and test activities. The validation techniques used are based on complex simulations of models building on the standards in the considered domain.

## 3.2. Model Driven Engineering for Distributed Software

**Keywords:** *Objects*, *UML*, *aspects*, *contracts*, *design patterns*, *models*, *product lines*, *software components*.

### *3.2.1. Software Product Lines*

It is seldom the case nowadays that we can any longer deliver software systems with the assumption that one-size-fits-all. We have to handle many variants accounting not only for differences in product functionalities (range of products to be marketed at different prices), but also for differences in hardware (e.g.; graphic cards, display capacities, input devices), operating systems, localization, user preferences for GUI ("skins"). Obvioulsy, we do not want to develop from scratch and independantly all of the variants the marketing department wants. Furthermore, all of these variant may have many successive versions, leading to a two-dimensional vision of product-lines.

### *3.2.2. Object-Oriented Software Engineering*

The object-oriented approach is now widespread for the analysis, the design, and the implementation of software systems. Rooted in the idea of modeling (through its origin in Simula), object-oriented analysis, design and implementation takes into account the incremental, iterative and evolutive nature of software development [57], [54]: large software system are seldom developed from scratch, and maintenance activities represent a large share of the overall development effort.

In the object-oriented standard approach, objects are instances of classes. A class encapsulates a single abstraction in a modular way. A class is both *closed*, in the sense that it can be readily instanciated and used by clients objects, and *open*, that is subject to extensions through inheritance [59].

### *3.2.3. Design Pattern*

Since by definition objects are simple to design and understand, complexity in an object-oriented system is well known to be in the *collaboration* between objects, and large systems cannot be understood at the level of classes and objects. Still these complex collaborations are made of recurring patterns, called design patterns. The idea of systematically identifying and documenting design patterns as autonomous entities was born in the late 80's. It was brought into the mainstream by such people as Beck, Ward, Coplien, Booch, Kerth, Johnson, etc. (known as the Hillside Group). However the main event in this emerging field was the publication, in 1995, of the book *Design Patterns: Elements of Reusable Object Oriented Software* by the so-called Gang of Four (GoF), that is Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides [56]. Today, design patterns are widely accepted as useful tools for guiding and documenting the design of object-oriented software systems. Design patterns play many roles in the development process. They provide a common vocabulary for design, they reduce system complexity by naming and defining abstractions, they constitute a base of experience for building reusable software, and they act as building blocks from which more complex designs can be built. Design patterns can be considered reusable micro-architectures that contribute to an overall system architecture. Ideally, they capture the intent behind a design by identifying the component objects, their collaborations, and the distribution of responsibilities. One of the challenges addressed in the Triskell project is to develop concepts and tools to allow their formal description and their automatic application.

### 3.2.4. Component

The object concept also provides the bases needed to develop *software components*, for which Szyperski's definition [62] is now generally accepted, at least in the industry:

> *A software component is a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third party.*

Component based software relies on assemblies of components. Such assemblies rely in turn on fundamental mechanisms such as precise definitions of the mutual responsability of partner components, interaction means between components and their non-component environment and runtime support (e.g. .Net, EJB, Corba Component Model).

Components help reducing costs by allowing reuse of application frameworks and components instead of redeveloping applications from scratch (product line approach). But more important, components offer the possibility to radically change the behaviors and services offered by an application by substitution or addition of new components, even a long time after deployment. This has a major impact of software lifecycle, which should now handle activities such as:

- design of component frameworks,
- design of reusable components as deployment units,
- validation of component compositions coming from various origins,
- component life-cycle management.

Empirical methods without real component composition models have appeared during the emergence of a real component industry (at least in the Windows world). These methods are now clearly the cause of untractable validation and of integration problems that can not be transposed to more critical systems (see for example the accidental destruction of Ariane 501 [58]).

Providing solutions for formal component composition models and for verifiable quality (notion of *trusted components*) are especially relevant challenges. Also the methodological impact of component-based development (for example within the maturity model defined by the SEI (CMM model)) is also worth attention.

### 3.2.5. Contracts

Central to this trusted component notion is the idea of *contract*. A software contract captures mutual requirements and benefits among stake-holder components, for example between the client of a service and its suppliers (including subcomponents). Contracts strengthen and deepen interface specifications. Along the lines of abstract data type theory, a common way of specifying software contracts is to use boolean assertions called pre- and post-conditions for each service offered, as well as class invariants for defining general consistency properties. Then the contract reads as follows: the client should only ask a supplier for a service in a state where the class invariant and the precondition of the service are respected. In return, the supplier promises that the work specified in the postcondition will be done, and the class invariant is still respected. In this way rights and obligations of both client and supplier are clearly delineated, along with their responsibilities. This idea was first implemented in the Eiffel language [60] under the name *Design by Contract*, and is now available with a range of expressive power into several other programming languages (such as Java) and even in the Unified Modeling Language (UML) with the Object Constraint Language (OCL) [63]. However, the classical predicate based contracts are not enough to describe the requirements of modern applications. Those applications are distributed, interactive and they rely on resources with random quality of service. We have shown that classical contracts can be extended to take care of synchronization and extrafunctional properties of services (such as throughput, delays, etc) [53].

### 3.2.6. Models and Aspects

As in other sciences, we are increasingly resorting to modelling to master the complexity of modern software development. According to Jeff Rothenberg, *Modeling, in the broadest sense, is the cost-effective use of something in place of something else for some cognitive purpose. It allows us to use something that is simpler, safer or cheaper than reality instead of reality for some purpose. A model represents reality for the given purpose; the model is an abstraction of reality in the sense that it cannot represent all aspects of reality. This allows us to deal with the world in a simplified manner, avoiding the complexity, danger and irreversibility of reality.*

So modeling is not just about expressing a solution at a higher abstraction level than code. This has been useful in the past (assembly languages abstracting away from machine code, 3GL abstracting over assembly languages, etc.) and it is still useful today to get a holistic view on a large C++ program. But modeling goes well beyond that.

Modeling is indeed one of the touchstone of any scientific activity (along with validating models with respect to experiments carried out in the real world). Note by the way that the specificity of engineering is that engineers build models of artefacts that usually do not exist yet (with the ultimate goal of building them).

In engineering, one wants to break down a complex system into as many models as needed in order to address all the relevant concerns in such a way that they become understandable enough. These models may be expressed with a general purpose modeling language such as the Unified Modeling Language (UML), or with Domain Specific Languages when it is more appropriate.

Each of these models can be seen as the abstraction of an aspect of reality for handling a given concern. The provision of effective means for handling such concerns makes it possible to establish critical trade-offs early on in the software life cycle, and to effectively manage variation points in the case of product-lines.

Note that in the Aspect Oriented Programming community, the notion of aspect is defined in a sligthly more restricted way as the modularization of a cross-cutting concern. If we indeed have an already existing “main” decomposition paradigm (such as object orientation), there are many classes of concerns for which clear allocation into modules is not possible (hence the name “cross-cutting”). Examples include both allocating responsibility for providing certain kinds of functionality (such as loggin) in a cohesive, loosely coupled fashion, as well as handling many non-functional requirements that are inherently cross-cutting e.g.; security, mobility, availability, distribution, resource management and real-time constraints.

However now that aspects become also popular outside of the mere programming world [61], there is a growing acceptance for a wider definition where an aspect is a concern that can be modularized. The motivation of these efforts is the systematic identification, modularization, representation, and composition of these concerns, with the ultimate goal of improving our ability to reason about the problem domain and the corresponding solution, reducing the size of software model and application code, development costs and maintenance time.

### *3.2.7. Design and Aspect Weaving*

So really modeling is the activity of separating concerns in the problem domain, an activity also called *analysis*. If solutions to these concerns can be described as aspects, the design process can then be characterized as a weaving of these aspects into a detailed design model (also called the solution space). This is not new: this is actually what designers have been effectively doing forever. Most often however, the various aspects are not *explicit*, or when there are, it is in the form of informal descriptions. So the task of the designer is to do the weaving in her head more or less at once, and then produce the resulting detailled design as a big tangled program (even if one decomposition paradaigm, such as functional or object-oriented, is used). While it works pretty well for small problems, it can become a major headache for bigger ones.

Note that the real challenge here is not on how to design the system to take a particular aspect into account: there is a huge design know-how in industry for that, often captured in the form of Design Patterns (see above). Taking into account more than one aspect as the same time is a little bit more tricky, but many large scale successful projects in industry are there to show us that engineers do ultimately manage to sort it out.

The real challenge in a product-line context is that the engineer wants to be able to change her mind on which version of which variant of any particular aspect she wants in the system. And she wants to do it cheaply, quickly and safely. For that, redoing by hand the tedious weaving of every aspect is not an option.

### *3.2.8. Model Driven Engineering*

Usually in science, a model has a different nature that the thing it models ("do not take the map for the reality" as Sun Tse put it many centuries ago). Only in software and in linguistics a model has the same nature as the thing it models. In software at least, this opens the possibility to automatically derive software from its model. This property is well known from any compiler writer (and others), but it was recently be made quite popular with an OMG initiative called the Model Driven Architecture (MDA). This requires that models are no longer informal, and that the weaving process is itself described as a program (which is as a matter of facts an executable meta-model) manipulating these models to produce a detailled design that can ultimately be transformed to code or at least test suites.

The OMG has built a meta-data management framework to support the MDA. It is mainly based on a unique M3 “meta-meta-model” called the Meta-Object Facility (MOF) and a library of M2 meta-models, such as the UML (or SPEM for software process engineering), in which the user can base his M1 model.

The MDA core idea is that it should be possible to capitalize on platform-independent models (PIM), and more or less automatically derive platform-specific models (PSM) –and ultimately code– from PIM through model transformations. But in some business areas involving fault-tolerant, distributed real-time computations, there is a growing concern that the added value of a company not only lies in its know-how of the business domain (the PIM) but also in the design know-how needed to make these systems work in the field (the transformation to go from PIM to PSM). Reasons making it complex to go from a simple and stable business model to a complex implementation include:

- Various modeling languages used beyond UML,
- As many points of views as stakeholders,
- Deliver software for (many) variants of a platform,
- Heterogeneity is the rule,
- Reuse technical solutions across large product lines (e.g. fault tolerance, security, etc.),
- Customize generic transformations,
- Compose reusable transformations,
- Evolve and maintain transformations for 15+ years.

This wider context is now known as Model Driven Engineering.

## 3.3. Mathematical Foundations for Distributed and Reactive Software

**Keywords:** *Labeled transition systems*, *event structures*, *partial orders*.

### *3.3.1. Transition systems*

A labeled transition system (or LTS) is a directed graph which edges, called transitions, are labeled by letters from an alphabet of **events**. The vertices of this graph are called **states**. A LTS can be defines as a tuple $M = (Q^M, A, T^M \subset Q^M \times A \times Q^M, q^M_{init})$, in which $Q^M$ is a set of states, $q^M_{init}$ is an initial state, $A$ is a set of events, $T^M$ is a transition relation.

Note that from this definition, the set of states in a LTS s not necessarily finite. Usually, the term **finite state automata** is used to designate a LTS with a finite set of states and events. In fact, automatas are the simplest models than can be proposed. They are often used to model reactive (and usually distributed) systems. Within this framework, events represent the interactions (inputs and outputs) with the environment. The term input/output LTS (IO-LTS) is often used to designate this kind of automata.

Labeled transition systems are obtained from reactive systems specifications in high-level description languages such as UML. The construction of a LTS from a specification is done using an operational semantics for this language, which is usually formalized as a deduction rules system. For simple languages such as process algebras (like CCS ), operational semantics can be defined using less than axioms and inference rules, while for notations such as UML, semantics would be defined in more than 100 pages.

For performance reasons, these operational semantics rules are never used directly, and are subject to several transformations. For example, the way states are encoded is an efficiency factor for LTS generation.

Computation of transformations of LTS can be resumed to search and fix-point calculus on graphs. These calculi can be performed either explicitely or implicitly, without an exhaustive calculus or storage of a LTS.

Classical algorithms in language theory build explicitely finite state automatas, that are usually integrally stored in memory. However, for most of the problems we are interested in, exhaustive construction or storage of an LTS is not mandatory. Partial construction of an LTS is enough, and strategies similar to lazy evaluation in functional programs can be used: the only part of LTS computed is the part needed for the algorithm.

Similarly, one can forget a part of a LTS previously computed, and hence recycle and save memory space. The combination of these implicit calculus strategies allow the study of real size systems even on reasonably powerful machines.

### 3.3.2. Non interleaved models

One of the well known drawbacks of LTSs [55] is that concurrency is represented by means of behaviors interleaving. This is why LTS, automatas and so on are called “interleaved models”. With interleaved models, a lot of memory is lost, and models represented can become very complex. Partial order models partially solve these problems.

A partial order is a tuple $(E, \leq, \Pi, \varphi, \Sigma, I)$ in which:

- $E$ represents a set of atomic events, that can be observable or not. Each event is the occurrence of an action or operation. It is usually considered that an event is executed by an unique process in an system.
- $\leq$ is a partial order relation that describes a precedence relation between events. This order relation can be obtained using the hypotheses that:
  1. processes are sequential : two events executed by the same process are causally ordered.
  2. communications are asynchronous and point to point: the emission of a message precedes its reception.
- $\sigma$ is an alphabet of actions.
- $I$ is a set of process names
- $\Pi : \Sigma \to I$ is an action placement function.
- $\varphi : E \to \Sigma$ is an event labeling function

A partial order can be used to represent a set of executions of a system in a more “compact” way than interleaved models. Another advantage of partial order models is to represent explicitely concurrency : two events that are not causally dependant can be executed concurrently. In a LTS, such a situation would have been represented by an interleaving.

A linearization of a partial order is a total order that respect the causal order. Any linearization of a partial order is a potential execution of the system represented. However, even if partial order can represent several executions, linearizations do not represent a real alternative. This problem is solved by a more complete partial order model called event structures.

A prime *event structure* [64] is a partial order equipped with an additional binary conflict relation. An event structure is usually defined by a tuple $(E, \leq, \sharp, \Pi, \varphi, \Sigma, I)$ where:

- $E, \leq, \Pi, \varphi, \Sigma, I$ have the same signification as previously,
- $\sharp \subseteq E \times E$ is a binary and symmetric relation that is inherited through causality $(\forall e \sharp e', e \leq e'' \implies e'' \sharp e')$.

The conflict relation of an event structure defines pairs of events that can not appear in the same execution of the system represented, hence introducing alternative in partial orders. The potential executions of a system represented by an event structures are linearizations of conflict free orders contained in the structure. The main advantage of event structures is to represent at the same time concurrency and alternative in a partial order model. We think that these models are closer to human understanding of distributed systems executions than interleaved models.

# 4. Application Domains

## 4.1. Software for Telecommunication and Real-Time Systems

**Keywords:** *UML*, *distributed systems*, *software engineering*, *telecommunication*, *test*.

In large scaled distributed systems such as developed for telecommunications, building a new application from scratch is no longer possible. There is a real need for flexible solutions allowing to deal at the same time with a wide range of needs (product lines modeling and methodologies for managing them), while reducing the time to market (such as derivation and validation tools).

Triskell has gained experience in model engineering, and finds here a propitious domain. The increasing software complexity and the reliability and reusability requirements fully justify the methods developed by our project. The main themes studied are reliable software components composition, UML-based developments validation, and test generation from UML models, iether at requirement level or at design level.

The research activity in Triskell focuses at the same time on development efficiency and reliability. Our main applications mainly concern reliable construction of large scale communicating software, and object oriented systems testing.

Reliability is an essential requirement in a context where a huge number of softwares (and sometimes several versions of the same program) may coexist in a large system. On one hand, software should be able to evolve very fast, as new features or services are frequently added to existing ones, but on the other hand, the occurrence of a fault in a system can be very costly, and time consuming. A lot of attention should then be paid to interoperability, *i.e. the ability for software to work properly with other.* We think that formal methods may help solving this kind of problems. Note that formal methods should be more and more integrated in an approach allowing system designer to build software globally, in order to take into account constraints and objectives coming from user requirements.

Software testing is another aspect of reliable development. Testing activities mainly consist in ensuring that a system implementation conforms to its specifications. Whatever the efforts spent for development, this phase is of real importance to ensure that a system behaves properly in a complex environment. We also put a particular emphasis on on-line approaches, in which test and observation are dynamically computed during execution.

# 5. Software

## 5.1. Kermeta : Kernel Metamodeling

**Keywords:** *MDA*, *MOF*, *UML*, *model transformation*.

**Participants:** Olivier Barais, Franck Chauvel, Zoé Drey, Franck Fleurey, Cyril Faucher, Jean-Marc Jézéquel, Jean-Marie Mottu, Pierre-Alain Muller, Jim Steel, David Touzet, Didier Vojtisek [correspondant].

Nowadays, object-oriented meta-languages such as MOF (Meta-Object Facility) are increasingly used to specify domain-specific languages in the model-driven engineering community. However, these meta-languages focus on structural specifications and have no built-in support for specifications of operational semantics. Triskell has developped the Kermeta language to explore the idea of using aspect-oriented modeling to add precise action specifications with static type checking and genericity at the meta level, and examine related issues and possible solutions [35].

Kermeta consists of an extension to the Essential Meta-Object Facilities (EMOF) 2.0 to support behavior definition. It provides an action language to specify the body of operations in metamodels. This action language is imperative and object-oriented.

Kermeta is used in several use cases:

- to give a precise semantic of the behavior of a metamodel which then can be simulated.
- to act as a model transformation language.
- to act as a constraint language.

The development environment built for the Kermeta language currently provides the following tools

- an interpreter that allows a metamodel to be executed.
- text and graphical editors, fully integrated within Eclipse, with syntax higlighting, code autocompletion.
- an Eclipse outline view, which allows navigation through the whole model and metamodel.
- various import/export transformations such as ecore2kermeta (kermeta text), kermat2ecore, kermeta2xmi (xmi version of your kermeta metamodel), xmi2kermeta, xmi2ecore.

Developped as an open source software under the terms of the EPL (Eclipse Public License), it has been first deposited to the APP (Agence de Protection des Programmes) in October 2005.

Thanks to Kermeta it is possible to build various frameworks dedicated to domain specific metamodels. Those frameworks are organised into MDKs (Model Development Kits). For example, Triskell porposes MDKs to work with the following metamodels: Java5, UML2, RDL (requirements), Ecore, Traceability, ...Some of these MDKs (UML2, RDL) are advanced enough to constitute a complete application.

## 5.2. UMLAUT NG : Extendible model transformation tool and framework

**Keywords:** *MDA*, *MOF*, *UML*, *component*, *model transformation*, *patterns*, *validation*.

**Participants:** Franck Chauvel, Franck Fleurey, Jean-Marc Jézéquel, Vincent Mahé, Damien Pollet, Didier Vojtisek [correspondant].

UMLAUT is a model tranformation framework for UML (*Unified Modelling Language*) developped since 1998 by Triskell. It allows to apply complex transformations to UML models. UMLAUT NG is the evolution of this framework in order completly take into account the latest versions of UML metamodel and fit to the global approach of the team. It is now distributed as a Kermeta MDK. Thanks to the operationnal semantic expressed using Kermeta, this allows to improve the correctness of the model by applying one or many of the available methods. This includes the simulation of the user model, the use of validation tools like the one from the CADP toolbox, the application of complex transformations, the generation code, etc.

## 5.3. Sintaks : Textual syntaxes for models

**Keywords:** *MDA*, *MOF*, *UML*, *model transformation*, *syntax*.

**Participants:** Pankaj Bhatia, Erwan Brottier, Pierre-Alain Muller, David Touzet, Didier Vojtisek [correspondant].

The Sintaks tool enables to define bridges between concrete (textual files) and abstract syntax (models). It automates the process to build parser and pretty printer that are typically used by textual editors.

A bridge consists in a Sintaks model that defines the way to:

- parse a text in order to get the corresponding model (with respect to a given metamodel);
- explore a model in order to pretty print its textual representation.

Sintaks is based onto the EMF repository and then is compatible with most of the modeling tools of the MDA community running in Eclipse.

## 5.4. Requested : a toolbox for requirement simulation and testing

**Keywords:** *Test*, *requirement simulation*, *requirement testing*, *textual requirements*, *use cases*.

**Participants:** Benoit Baudry [correspondant], Yves Le Traon, Franck Fleurey, David Touzet.

The objective of the Requested toolbox is to offer a MDA transformation from textual requirements to simulable requirements within the UML (use cases + scenarios). It allows the simulation of requirements and the automated generation of test objectives. Two tools are under development:

1. The transformation of natural language requirements expressed in the RDL language (Requirement Description Language) into a use case model, enhanced with contracts.
2. The UCTS system allows the simulation of the use case model, enhanced with contracts, and the automated generation of test objective.

More precisely, UCTSystem is a prototype designed to perform automatic test generation from UML requirements. It uses UML use cases enhenced with contracts (*i.e. precondition and postconditions*) to build an execution model allowing all valid sequences of use cases. Using this execution model and several test criteria, it generates test objectives as sequence of use cases to exerce. It includes both criteria for functional testing and a criterion for robusness testing. Those test objectives are then mapped into test cases using test templates.

# 6. New Results

## 6.1. Contract-based and Aspect Oriented Design

### *6.1.1. Behavioural Aspects and Weaving*

**Participants:** Jacques Klein, Jean-Marc Jézéquel, Franck Fleurey.

The idea of encapsulating crosscutting concerns into the notion of aspects looks very promising for complementing the usual notion of modules available in most languages. By localizing these crosscutting concerns, the software engineer can get a better control over variations, either in the spatial dimension (product line context) or in the temporal dimension (software evolutions). The need to isolate these crosscutting concerns has been popularized by the Aspect-J programming language, but there is a growing interest in also handling them earlier in the software life-cycle, for instance at design time, or during requirements analysis. With this intention, our work [13], [37], [38] (done in collaboration with Loic Helouet from the Distribcom project), proposes a weaving process for scenarios which is two-phased. Firstly a detection phase searches parts in a base model of scenarios. Secondly the composition phase builds a woven model of scenarios by composing the advice into the base model for each detected part. Especially, in [13], [37], we propose various interpretations for pointcuts that allow multiple behavioural aspects to be statically woven into finite scenarios. The idea is to allow join points to match a pointcut even when some extra-messages occur in between. Moreover, in [13], [38], we propose a semantic-based aspect weaving algorithm for infinite scenarios. The algorithm proposed uses a set of transformations that take into account the compositional semantics of scenarios (HMSCs) to weave an initial HMSC and a behavioral aspect expressed with scenarios. Finally, in [13], [37], the composition phase builds a woven model of scenarios by composing the advice into the base model for each detected part. The entire weaving process is automated and is implemented as model transformations within the Kermeta environment.

### *6.1.2. Managing Concerns in Software Architectures*

**Participants:** Olivier Barais, Jean-Marc Jézéquel, Noel Plouzeau, Sébastien Saudrais.

In paper [44] we propose a software development approach that emphasizes support for time properties in software components. Our software process is best suited to developers that (1) build applications from components with timing and performance concerns and/or (2) design software components for this kind of applications. The design process covers the lifecycle of component based design: (1) specification of time properties for services, (2) specification of component types, (3) specification of the behavioral and time implementation of a component, (4) composition of components and verification of the composition's properties, (5) generation of observers and of an observation framework using the Giotto runtime and semantics of execution.

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

Software architectures must frequently evolve to cope with changing requirements, and this evolution often implies integrating new concerns. Unfortunately, existing architecture description languages provide little or no support for this kind of evolution. The software architect must modify the architecture manually, which risks introducing inconsistencies. In previous work, we have proposed the TranSAT framework, which provides a pattern construct for describing new concerns and their integration into an existing architecture. As the interaction between the new concern and the existing architecture may be complex, it is essential that the framework ensure the coherence of the resulting architecture. The result of this work is the definition of a language for specifying patterns and verifications that ensure that the concern represented by a pattern can be safely integrated into an existing architecture. In [26], we present the verifications which comprise static verifications that check coherence properties before the architecture is modified and dynamic verifications that focus on the parts of the architecture that are affected by the pattern. As a result of these verifications, patterns can be provided as a commodity, such that a software architect can confidently apply a pattern obtained from a third-party developer. In [46], we present the implementation of the TranSAT framework. Precisely, this work investigates the use of a rule engine based on the Rete algorithms to implement the static analyzer, and the logic pointcut language of TranSAT designed at the software architectural level. These works have been carried out in collaboration with the Jacquard project (LIFL) in the context of the AOSD-Europe NoE).

## 6.2. Model-Based Testing

### *6.2.1. Design by Contract to improve Software Vigilance*

**Participants:** Yves Le Traon, Benoit Baudry, Jean-Marc Jézéquel.

Design by Contract is a lightweight technique for embedding elements of formal specification (such as invariants, pre- and post-conditions) into an object-oriented design. When contracts are made executable, they can play the role of embedded, on-line oracles. Executable contracts allow components to be responsive to erroneous states, and thus may help in detecting and locating faults. In this work [18], we define Vigilance as the degree to which a program is able to detect an erroneous state at runtime. Diagnosability represents the effort needed to locate a fault once it has been detected. In order to estimate the benefit of using Design by Contract, we formalize both notions of Vigilance and Diagnosability as software quality measures. The main steps of measure elaboration are given, from informal definitions of the factors to be measured to the mathematical model of the measures. As is the standard in this domain, the parameters are then fixed through actual measures, based on a mutation analysis in our case. Several measures are presented that reveal and estimate the contribution of contracts to the overall quality of a system in terms of vigilance and diagnosability.

### *6.2.2. Metamodel-based Model Synthesis for Model Transformations Testing*

**Participants:** Franck Fleurey, Jim Steel, Sagar Sen, Erwan Brottier, Benoit Baudry, Yves Le Traon.

In a Model-Driven Development context (MDE), model transformations allow memorizing and reusing design know-how, and thus automate parts of the design and refinement steps of a software development process. A model transformation program is a specific program, in the sense it manipulates models as main parameters. Each model must be an instance of a "metamodel", a metamodel being the specification of a set of models. Programming a model transformation is a difficult and error-prone task, since the manipulated data are clearly complex. In a first work [29], we have proposed a deterministic algorithm to generate input test data (called test models) for model transformations. However, due to a large number of decisions that have to be made during generation, this deterministic approach has some limitations. We thus have started investigating an evolutionnary approach to deal with these issues [45].

### 6.2.3. *Multi-Language Support for Model-Driven Requirement Analysis and Test Generation*

**Participants:** Benoit Baudry, Waqas Ahmed Saeed.

Expressing a valid requirements model is a crucial activity in a MDE context as it provides the starting point for further refinement steps that will lead to the implementation. In previous work we have proposed a requirements model from which it is possible to do simulation and high-level test generation. A requirements metamodel captures the key concepts to model the requirements and allows us to define several automatic model transformations to manipulate the requirements. In this work [42] (carried out in collaboration with Clémentine Nebut from LIRM), we focus on surface languages that can assist the edition of such a requirements model for simulation or testing purposes. We propose to use two different languages: a constrained natural language and UML activity diagrams. For each language, we discuss the main interesting features to express requirements and we define a model transformation to generate the corresponding requirements model.

### 6.2.4. *Towards Testable and Executable Themes*

**Participants:** Benoit Baudry, Jacques Klein.

Design validation is important for detecting errors early in the development life cycle. Testing the design is one significant means to achieve design validation. In this paper we introduce the KerTheme model. KerTheme provides a means for symmetrically decomposing concern based executable class diagrams and concern test scenarios. KerTheme also facilitates synchronised merging of these decomposed models into a coherent composite concern based executable class model and corresponding test scenarios. In these works [34], [33] (carried out in collaboration with Andrew Jackson and Siobhan Clarke from Trinity College, Dublin in the context of the AOSD-Europe NoE), we investigate whether decomposed concern based executable class diagrams simplifies the definition of concern test scenarios. This also allows us to investigate whether this approach ensures more rigorous testing of a complete system.

### 6.2.5. *Improving Test Suites for Efficient Fault Localization*

**Participants:** Benoit Baudry, Franck Fleurey, Yves Le Traon.

The need for testing-for-diagnosis strategies has been identified for a long time, but the explicit link from testing to diagnosis (fault localization) is rare. Analyzing the type of information needed for efficient fault localization, this work [28] identifies the attribute (called Dynamic Basic Block) that restricts the accuracy of a diagnosis algorithm. Based on this attribute, a test-for-diagnosis criterion is proposed and validated through rigorous case studies: it shows that a test suite can be improved to reach a high level of diagnosis accuracy. So, the dilemma between a reduced testing effort (with as few test cases as possible) and the diagnosis accuracy (that needs as much test cases as possible to get more information) is partly solved by selecting test cases that are dedicated to diagnosis [28].

## 6.3. Model-Driven Engineering

### 6.3.1. *Model Typing*

**Participants:** Jim Steel, Jean-Marc Jézéquel.

Where object-oriented languages deal with objects as described by classes, model-driven development uses models, as graphs of interconnected objects, described by metamodels. A number of new languages have been and continue to be developed for this model-based paradigm, both for model transformation and for general programming using models. Many of these use single-object approaches to typing, derived from solutions found in object-oriented systems, while others use metamodels as model types, but without a clear notion of polymorphism. Both of these approaches lead to brittle and overly restrictive reuse characteristics. In this work [24], we propose a simple extension to object-oriented typing to better cater for a model-oriented context, including a simple strategy for typing models as a collection of interconnected objects. We suggest extensions to existing type system formalisms to support these concepts and their manipulation. We show using this system and motivating examples how this extended approach permits more flexible reuse, while preserving type safety.

### *6.3.2. Towards a Generic and Extensible Merge Operator*

**Participants:** Olivier Barais, Jean-Marc Jézéquel, Franck Fleurey.

Merging is a common way to compose both crosscutting and noncrosscutting concerns in a Model Driven software development. In this work [32] (carried out in collaboration with Andrew Jackson and Siobhan Clarke from Trinity College, Dublin in the context of the AOSD-Europe NoE), we argue that merge can be defined more generically as an operator at the meta-modelling level. By describing merge at this level, a merge operator can be used to compose models based on meta-models other than UML. There are various merge variants and we concede that a full unification of all merge semantics may be infeasible. To define a common merge, we propose the definition of a common merge kernel as a semantic base that can be extended to realise the different expressions of merge. This work can be used as a based for the KerTheme approach [34], [33] to express the composition of base themes and to express the composition of an aspect theme with a KerTheme. A base theme models a noncrosscutting concern and an aspect theme models a crosscutting concern.

#### *6.3.2.1. Reusable MDA Components: a Testing-for-Trust Approach Based on Mutation Analysis*

**Participants:** Jean-Marie Mottu, Franck Fleurey, Robert France, Sudipto Ghosh, Benoit Baudry, Yves Le Traon.

Model transformations play a critical role in Model Driven Engineering, and thus efficient techniques for testing model transformations testing are needed [27]. Making model transformations trustable is an obvious target for model-driven development since they impact on the design process reliability. Ideally, model transformations should be designed and tested so that they may be used and reused safely as MDA components. The result of this work is a method for building trustable MDA components [40]. We first define the notion of MDA component as composed of its specification, one implementation and a set of associated test cases. The testing-for-trust approach checks the consistency between these three facets using the mutation analysis. Mutation analysis is an efficient technique to evaluate the quality of test data, and has been extensively studied both for procedural and object-oriented languages. We study how it can be adapted to model oriented programming [39]. Since no model transformation language has been widely accepted today, we propose generic fault models that are related to the model transformation process. First, we identify abstract operations that constitute this process: model navigation, models elements filtering, output model creation and input model modification. Then, we propose a set of specific mutation operators which are directly inspired from these operations. We believe that these operators are meaningful since a large part of the errors in a transformation are due to the manipulation of complex models regardless of the concrete implementation language. Used in the testing-for-trust approach, mutation analysis points out the lack of efficiency of the tests and the lack of precision of the specification. The mutation score produced evaluates: the level of consistency between the components facets and the level of trust we can have in a component. Relying on this estimation of the component trustability, developers can consciously trade reliability for resources.

#### *6.3.2.2. Reverse-engineering of UML 2.0 Sequence Diagrams from Execution Traces*

**Participants:** Romain Delamare, Benoit Baudry, Yves Le Traon.

To fully understand the behavior of a program, it is crucial to have efficient techniques to reverse dynamic views of the program. In this work [31], we focus on the reverse engineering of UML 2.0 sequence diagrams showing loops and alternatives from execution traces. To build these complete sequence diagrams, we need to capture the system's state through dynamic analysis. We build state vectors through trace analysis and we precisely discuss how the state of an object-oriented system can be captured. We also develop an adaptable trace analysis tool to experiment the ideas presented in this work.

# 7. Contracts and Grants with Industry

## 7.1. SPEEDS (IST)

**Keywords:** *COTS*, *SysML*, *UML*, *embedded systems*, *methods*, *system engineering*.

**Participants:** Jean-Marc Jézéquel, Mark Skipper, Olivier Barais.

SPEEDS is an IST Integrated Project defining the new generation of end-to-end methodologies, processes and supporting tools for safety-critical embedded system design. They will enable European systems industry to evolve from model-based design of hardware/software systems, towards integrated component based construction of complete virtual system models.

SPEEDS aims at improving substantially the competitiveness of the European industry in this critical economic sector by marrying design competence with deep technical insights and theoretical foundations. SPEEDS partners are companies active in the entire supply chain: OEMs, suppliers, and tool vendors, supported by leading European research institutions. The technical pillars of the SPEEDS approach are:

- A semantics-based modeling method to
  - support the construction of complex embedded systems by composing heterogeneous subsystems.
  - enable sound integration of new and existing tools.

  This modeling approach defines "rich-component" models to represent both functional and non-functional aspects so that efficient implementations can be derived from abstract models.
- Novel formal analysis tools and techniques to assess precisely properties of the system that will allow to explore architectural alternatives of implementation platforms and enable correct-by-construction designs. Compositionality and abstractions will make this approach scalable for large systems.
- A new tool-supported process, controlled speculative design, minimizing the risk of concurrent design activities by establishing formal "contracts" between inter- and intracompany design groups.

European automotive, aeronautic and other leading European industrial sectors rely critically on embedded systems. These sectors produce high end products characterized by high performance, high dependability, outstanding quality demands, and exponentially increasing complexity. They will maintain leading market only if significant reductions in costs and development times can be achieved while maintaining high quality standards.

To sustain European competitiveness, SPEEDS launches a concerted effort, integrating European expertise in system design approaches and formal techniques for quality assessments, as well as key industrial players and leading tool vendors to define the new generation of end-to-end system design processes. This will enable European systems industry to evolve from model-based design of embedded systems, towards integrated component based construction of complete virtual system models. It will include:

- Construction of early system prototypes during the design stage.
- Thorough quality and stability assessment at early design stages.
- Active treatment of design assumptions to guide system development.
- Support for concurrent multi-organization development of complex designs.

SPEEDS aims to achieve improvements in re-use reductions in integration time and change processing time and reductions in the number of significant design iterations.

An additional expected result is the establishment of a standard for embedded systems modeling and model-based design process across European industry. SPEEDS will perform key innovations in architectural abstractions capturing functional and non-functional features, in formal modelling (semantic based integration of heterogeneous system models with component models covering all non-functional constraints), and in formal multi-viewpoint analysis covering functional, timeliness, safety, and dependability requirements performed across all system design levels.

All capabilities are offered through the SPEEDS engineering environment which provides seamless integration of SPEEDS analyses and design capabilities with existing COTS tools for system engineering, allowing the engineers to benefit from SPEEDS innovations in their established tool environment.

The SPEEDS design process will focus on re-use of design components at all design levels, design space exploration and early system-analysis through virtual integration. It will be based on an innovative "speculative design" approach of tracking design uncertainties while supporting concurrent multi-organizational development of complex embedded systems.

Currently the Triskell team participate mainly to the SP.2 work package named heterogeneous rich components (HRC) to define a semantic-based common meta-model, Which forms the foundations for the component based construction of complete virtual system models.

Project duration: 2006-2009

Triskell budget share: 201 keuros

Project Coordinator: Airbus

Participants: Airbus Deutschland GmbH (A-D), Airbus France S.A.S. (A-F), DaimlerChrysler AG (DC), Israel Aircraft Industries Ltd (IAI), Robert Bosch GmbH, INRIA, Kuratorium OFFIS e.V., PARADES, Universite Joseph Fourier, TNI, I-Logix Israel Ltd, Extessy AG, Knorr Bremse Fekrendszerek Kft, Steyr GmbH & Co KG, SAAB AB, Esterel Technologies SA

## 7.2. AOSD-Europe (Network of Excellence)

**Keywords:** *Aspect Oriented Design*.

**Participants:** Jean-Marc Jézéquel, Noël Plouzeau, Yves Le Traon, Jacques Klein, Olivier Barais, Sébastien Saudrais, Didier Vojtisek.

Aspect-Oriented Software Development (AOSD) supports systematic identification, modularisation, representation and composition of crosscutting concerns such as security, mobility, distribution and resource management. Its potential benefits include improved ability to reason about the problem domain and corresponding solution; reduction in application code size, development costs and maintenance time; improved code reuse; architectural and design level reuse by separating non-functional concerns from key business domain logic; improved ability to engineer product lines; application adaptation in response to context information and better modelling methods across the lifecycle. AOSD-Europe will harmonise and integrate the research, training and dissemination activities of its members in order to address fragmentation of AOSD activities in Europe and strengthen innovation in areas such as aspect-oriented analysis and design, formal methods, languages, empirical studies and applications of AOSD techniques in ambient computing. Through this harmonisation, integration and development of essential competencies, the AOSD-Europe network of excellence aims to establish a premier virtual European research center on AOSD. The virtual research centre will synthesise the collective viewpoints, expertise, research agendas and commercial foci of its member organisations into a vision and pragmatic realisation of the application of AOSD technologies to improve fundamental quality attributes of software systems, especially those critical to the information society. It will also act as an interface and a centralised source of information for other national and international research groups, industrial organisations and governmental bodies to access the members' work and enter collaborative initiatives. The existence of such a premier research base will strengthen existing European excellence in the area, hence establishing Europe as a world leader.(http://www.aosd-europe.net/)

Project duration: 2004-2008

Triskell budget share: 150 keuros

Project Coordinator: University of Lancaster

Participants: University of Lancaster, Technical University of Darmstadt, INRIA, VUB, Trinity College Dublin, University of Malaga, Katholieke Universiteit Leuven, Technion, Siemens, IBM Hursley Development Laboratory

## 7.3. Artist2 (Network of Excellence)

**Keywords:** *Real-Time Component Models*.

**Participants:** Jean-Marc Jézéquel, Noël Plouzeau, Pierre-Alain Muller, Benoit Baudry, Didier Vojtisek.

The strategic objective of the ARTIST2 Network of Excellence is to strengthen European research in Embedded Systems Design, and promote the emergence of this new multi-disciplinary area. Artist2 gathers together the best European teams from the composing disciplines, and will work to forge a scientific community. Integration will be achieved around a Joint Programme of Activities, aiming to create critical mass from the selected European teams.

The ARTIST2 Network of Excellence on Embedded Systems Design is implementing an international and interdisciplinary fusion of effort to create a unique European virtual centre of excellence on Embedded Systems Design. This interdisciplinary effort in research is mandatory to establish Embedded Systems Design as a discipline, combining competencies from electrical engineering, computer science, applied mathematics, and control theory. The ambition is to compete on the same level as equivalent centres in the USA (Berkeley, Stanford, MIT, Carnegie Mellon), for both the production and transfer of knowledge and competencies, and for the impact on industrial innovation.

ARTIST2 has a double core, consisting of leading-edge research in embedded systems design issues (described later in this document) in the Joint Programme of Research Activities (JPRA), and complementary activities around shared platforms and staff mobility in the Joint Programme of Integration Activities (JPIA).

The JPRA activities are pure research, and the JPIA are complementary efforts for integration. Both work towards deep integration between the participating research teams.

The JPRA and JPIA are structured into clusters - one for each of the selected topics in embedded systems design (in red). Teams may be involved in one or several clusters.

Around this double core is the Joint Programme of Activities for Spreading Excellence (JPASE). These are complementary activiites for disseminating excellence across all available channels, targetting industry, students, and other European and international research teams.

Building the embedded systems design scientific community is an ambitious programme. To succeed, ARTIST2 will build on the achievements and experience from the ARTIST1 FP5 Accompanying Measure (http://www.artist-embedded.org/) on Advanced Real-Time Systems. ARTIST1 provided the opportunity to test the concept of a two-level integration (within and between clusters) four clusters in ARTIST2 originated as “actions” in ARTIST1. Building the ARTIST2 consortium and associated structure is the culmination of discussions and ambitions elaborated within ARTIST1.

ARTIST2 addresses the full range of challenges related to Embedded Systems Design, covering all aspects, ranging from theory through to applications. In this way, ARTIST2 is perfectly in line with the IST priority on embedded systems, and in particular with the focus area called “system design”.

The Triskell team is taking part in two Artist2 clusters: the *Real Time Components* cluster (led by Albert Benveniste, Irisa, and Bengt Jonsson, at Uppssala university, Sweden) and the Adaptive Real Time Middleware (led by Giorgo Buttazzo, Italy).

The current cooperation topics within the Real TimeComponents cluster are the use of various formalisms for timed behaviour descriptions, the definition of an architecture for interconnecting simulation and verification platforms for these behaviours. The Triskell team has designed a process and a tool chain to support specification, validation and monitoring of time issues in software components. This tool chain was implemented by integrating and extending existing tools from partners of the RTC cluster.

Within the Adaptive Real Time cluster, Triskell is participating in the common definition of quality of service dictionary, in the context of middleware runtimes. The Triskell project has also proposed a new metamodel for expressing quality of service properties of software components. The proposal is being compared and evaluated with respect to other metamodels proposed by Artist partners (including the Marte profile for UML proposed at OMG), in order to build a common Artist2 metamodel for quality of service.

Project duration: 2004-2008

Triskell budget share: 50 keuros

Project Coordinator: Verimag

Participants: see http://www.artist-embedded.org/artist/

## 7.4. UML profile for ground system (carroll)

**Keywords:** *AADL*, *MDA*, *OMG*, *UML*, *methodology*.

**Participants:** Jean-Marc Jézéquel, Pierre-Alain Muller, Didier Vojtisek.

"UML profile for ground system" is a lightweight project developed by CEA/LIST (LLSP), THALES Research and Technology, and INRIA acting as an expert group for CNES. This project aims at defining a UML profile for designing ground system in the aerospace domain.

Project duration: 2006-2007 years

Triskell budget share: 10 keuros

Project Coordinator: Thalès (TRT)

Participants: CEA, Thalès R&T INRIA Rennes, INRIA

## 7.5. Mopcom Hard (RNTL)

**Keywords:** *MDA*, *OMG*, *RT-E*, *UML*, *system on chip*.

**Participants:** Jean-Marc Jézéquel, Didier Vojtisek.

Mopcom hard is a project of the "pole de compétitivité" of britany. The project focuses on the use of model driven engineering for the development of embedded system typically based on system-on-chip (SOC). The project will produce a complete methodology and development environment dedicated to the domain.

In 2006, the main activity for the team was the mounting of the projects that will concretly start in 2007. Triskell participation will mainly apply to the specification of precise metamodels (using Kermeta) and of the development process.

Project duration: 2007-2010 years

Triskell budget share: 232 851 euros

Project Coordinator: Thalès (TSA)

Participants: Thalès Systèmes Aéroportés, Thomson, Sodius, ENSIETA, LESTER, Supelec Rennes, INRIA

## 7.6. Amadeus

**Keywords:** *MDA*, *MDE*, *UML*, *methodology*.

**Participants:** Jean-Marc Jézéquel, Pierre-Alain Muller, Didier Vojtisek.

Amadeus is a project supported by the PRIR of Bretagne Council. It involves ENSTBr, ENSIETA, Université de Bretagne Sud and Inria/Triskell. This project aims at building links between research teams in Brittany working on Model Driven Engineering. Its main scientific objectives are:

- study relationships between the notion of design by contract and model refinements,
- study formal projections of UML to help model verifications,
- apply a robust design and validation methodology to the UML

## 7.7. KEREVAL

**Keywords:** *components*, *diagnosis*, *extra functional testing*, *probes*.

**Participants:** Marouane Himdi, Yves Le Traon.

In addition to detection of errors related to design, coding or deployment of an application, the diagnosis is a well-known technique for understanding the behaviour of a software system and an absolute requirement for its improvement. Unfortunately, applications become more difficult to diagnosis as functionalities provided become complex. In collaboration with the KEREVAL company, we explore the use of dynamic probes (sensors) that will be injected into running system to collect various information. The innovative aspect of this approach is the use of generic probes to develop diagnosis framework.

## 7.8. OpenDevFactory

**Keywords:** *MDE*, *UML*, *metamodel*, *requirements engineering*, *traceability*.

**Participants:** Benoit Baudry, David Touzet, Erwan Brottier, Didier Vojtisek.

OpenDevFactory is a sub project of the project Usine Logicielle (labelled by the System@tic "pôle de compétitivité"). Its objective is to supply a standard platform for integrating technological developments for modelling software tools. This sub project produces technological components on top of which domain tools (automobile, security, telecommunication, aeronautical) can be derived at a lesser effort. That platform will be built as an interoperable federation of tools which limited parts could be deployed to make specialised IDEs meeting the particular needs of different kinds of users. The technological bricks are organized as follows:

- Technological infrastructure bricks for MDE such as providing support for model transformation, behaviour modelling as well as orchestration of engineering activities.
- Domain extension bricks supporting fault tolerance modelling, Real time embedded systems modelling, platforms modelling, requirements modelling or UML simulations.
- Integration technologies of MDE design environments with other engineering environments such as design environments for design of automatisms or critical embedded software.

The integration structure of OpenDevFactory is build on top of the Eclipse framework. On this structure, the generic and specific engineering components are case by case integrated depend-ing on the needs of the industrial projects case studies. In this context, Triskell has to develop an Eclipse plugin providing a requirements engineering integrated environment. This environment includes the following features:

- Requirements specification by means of a controled natural language (requirement description language).
- Definition of a requirements metamodel, and automated transformation from textual to model-based specifications.
- Definition of a usecase based metamodel encoding the dynamic semantics of the defined requirements.

- Parametrized interpretation (using interpretation patterns) of a requirements model in order to build its corresponding usecase model.
- Simultaion facilities enabled over the obtained usecase model.

Project duration: 2005-2008

Triskell budget share: 75 048 euros

Number of person/years: 43.2

Project Coordinator: Thales R&T

Participants: CEA, CS, Dassault Aviation, EADS, EDF, Esterel Technologies, Hispano Suiza, IFP, INRIA, LIP6, LRI, MBDA,Ecole Polytechnique, Softeam, Supelec, Thales, Trialog

## 7.9. DOMINO (RNTL)

**Keywords:** *domain specific languages*, *model transformation*, *model-driven engineering*, *reliability*, *validation*, *verification* .

**Participants:** Benoit Baudry, Jean-Marc Jézéquel, Jean-Marie Mottu, Yves Le Traon.

The DOMINO project (Methods and processes for domain specific modelling) is funded by the french agency for research (ANR). It aims at proposing a development process based on a mutli-view description of a system, each view being expressed with various domain specific modelling languages. Model-driven engineering is the core technology to define this process and is used to validate and verify the different artefacts produced at different steps of the process. A reliable process is crucial in the context of a multi-formalism approach to modelling. This process encompasses all the techniques needed to design, validate, and improve the software artefacts.

The Triskell develops techniques to validate and test model transformations that are used to automate different steps of the process. These techniques are based on model synthesis techniques for test input generation and on contracts to check the results of test cases. We also propose an incremental process to build and improve trust in model transformations that are encapsulated as reusable components.

Project duration: 2006-2008

Triskell budget share: 79 447,66 euros

Number of person/years: 21

Project Coordinator: IRIT

Participants: IRIT, Airbus, Sodifrance, CNES, CEA-LIST, ENSIETA, IRISA/Triskell

## 7.10. OpenEmbeDD (RNTL)

**Keywords:** *MDE*, *RT/E requirements engineering*, *RT/E system*, *formal proof*, *model transformation*, *model-checking*.

**Participants:** Pierre-Alain Muller, Jean-Marc Jézéquel, Didier Vojtisek, Cyril Faucher, Vincent Mahé, Zoé Drey, François Tanguy.

OpenEmbeDD is a project supported by the RNTL program. OpenEmbeDD is an Eclipse open-source platform, extensible, standardized and generic, based on the MDE approach for developing Real-Time and Embedded systems. OpenEmbeDD integrates the technologies based on formal models from synchronous/asynchronous/mixed paradigms. This platform covers the 2 branches of the V cycle : specification/design/implementation et checking/validation. The project takes part in the topic: embedded systems of RNTL. This project contributes to continue the effort of the RNTL for developing open-source software, a main topic of the RNTL. The building of the platform is in synergy with the "pôles de compétitivité" "SYSTEM@TIC-Paris Région" (Ile de France), "Aéronautique-Espace, Systèmes Embarqués" (Midi-Pyrénées) and "Images et Réseaux" (Bretagne). The platform is adopted in the research program CARROLL, this program is led for 2 years by the CEA, INRIA and THALES that are at the initiative of the OMG MARTE standard.

The main topics of the project are:

- Formal approach (abstraction, proof, model-checking, transformations).
- Modeling of Real-Time requirements.
- Modeling of Real-Time properties (components, systems,...).
- Process and tools for checking and validating (proof, tests,...).
- Languages and tools for describing and designing architectures.

The core of the platform is the metamodeling language Kermeta that is developed by the Triskell team. Kermeta is built on top of the Eclipse framework, it provides an interpreter and editors, in particular a graphical editor. In this context, Triskell has to develop an Eclipse plugin providing a graphical editor to design executable metamodels, a first version is available for the community. This editor is generated from the generator developed in the Topcased project (Airbus et al.). Triskell members participate at the specification of the source generator for building automatically graphical editors. We have studied the alignment between the compilers of ATL and of Kermeta. Finally, Triskell has setted the common platform for gathering and sharing all the tools.

Project duration: 2006-2009

total project budget: 7 Meuros

Trsikell budget share: 250 keuros

Number of person/years: 185.3

Project Coordinator: INRIA

Participants: Airbus, Anyware Technologies, CEA-List, CS, France Telecom, INRIA, LAAS, THALES (DAE and RT), Verimag

## 7.11. Faros (RNTL)

**Keywords:** *MDE*, *Web services*, *model transformation*, *quality of service*.

**Participants:** Noel Plouzeau, Jean-Marc Jézéquel, Franck Chauvel.

Faros is a project supported by the RNTL program. The Faros project has started in march, 2006. This project will last 36 months. The general objective of the project is the definition and the construction of a software process and tool chain to build reliable Web service based application. The process and its corresponding tool chain will be able to accept as input domain specific, platform independant components. The tool will generate platform specific implementations of these components, interconnected through Web services.

The general strategy of the process is based on model engineering. The project's workpackages are organized as follows:

1. definition of metamodels for managing business specific application description;
2. definition of metamodels for Web services platforms;
3. definition of a general metamodel to describe pivot models, which are business and platform independant;
4. definition of transformations to generate Web services implementation from business specific models, using automated model transformation techniques.

The project will use the applications of the industrial partners (France Telecom, Electricité de France and Alicante) as case studies to validate the process and its tool chain.

Within the Faros RNTL project, the Triskell project is responsible for the metamodelisation activity, the supervision of transformation designs and the production of the model transformation engine. More precisely, the core of the tool chain will be based on the Kermeta model transformation engine, which is being developed entirely by the Triskell team.

Project duration: 2006-2009

Project budget: 1 Meuros

Triskell budget share: 80 keuros

Project type: exploratory

Number of person/years: 30.8

Project Coordinator: France Telecom

Participants: France Telecom R&D, EDF R&D, Alicante (industrial partners), university of Nice (I3S laboratory), university of Rennes 1 (IRISA laboratory), university of Lille (LIFL laboratory)

# 8. Other Grants and Activities

## 8.1. International working groups

### *8.1.1. ERCIM Working Group on Software Evolution*

Numerous scientific studies of large-scale software systems have shown that the bulk of the total software-development cost is devoted to software maintenance. This is mainly due to the fact that software systems need to evolve continually to cope with ever-changing software requirements. Today, this is more than ever the case. Nevertheless, existing tools that try to provide support for evolution have many limitations. They are (programming) language dependent, not scalable, difficult to integrate with other tools, and they lack formal foundations.

The main goal of the proposed WG (http://w3.umh.ac.be/evol/) is to identify a set of formally-founded techniques and associated tools to support software developers with the common problems they encounter when evolving large and complex software systems. With this initiative, we plan to become a Virtual European Research and Training Centre on Software Evolution.

Triskell contributes to this working group on the following points:

- re-engineering and reverse engineering
- model-driven software engineering and model transformation
- impact analysis, effort estimation, cost prediction, evolution metrics
- traceability analysis and change propagation
- family and product-line engineering

### *8.1.2. CNRS GDRs*

The Triskell project is connected to the national academic community through a lightweight participation to several CNRS GDR (Groupement de Recherche).

- GDR ASR: Action IDM (on Model Driven Engineering) that has sponsored the Kermeta 2006 workshop (http://www.actionidm.org)
- GPL proposal - Génie de la Programmation et du Logiciel (http://www-lsr.imag.fr/GPL), where Jean-Marc Jézéquel is a member of the scientific committee.

### *8.1.3. Standardization at OMG*

Triskell project participates to normalization action at OMG (http://www.omg.org/):

- Triskell project participates to the RFP MOF2.0 QVT Query/view/Transformation. This RFP standardizes a model transformation language which is a key point in efficiently applying MDA.
- Triskell project participates to the new Executable UML foundation RFP. This RFP will standardize a subset of UML2.0 with a more precise execution semantic.
- Triskell project is also involved in other OMG groups which are related to the team interests. For example, it participates to the ORMSC group which formalizes the MDA approach, to the MDA user SIG which represents the end user point of view for MDA. It is also invloved in the more general Analysis and Design group which promotes standard modelling techniques including UML and MOF.
- Triskell initiated a wiki dedicated to share information about the OMG within the INRIA (http://omg.wiki.irisa.fr/).

### *8.1.4. Collaboration with foreign research groups*

- University of Oslo, Norway. Collaboration on the SWAT project (Semantics-preserving Weaving - Advancing the Technology) with Øystein Haugen and Birger Møller-Pedersen. The goal of this formal collaboration is to identify basic mechanisms behind the mechanisms we find in generics, aspect orientation, family modeling and generative programming, in general what mechanisms we should have in order to produce models/programs from generic models/programs or from fragments of models/programs.
- Colorado State University (CSU), USA. Collaboration on several issues related to model-driven development with Robert France and Sudipto Ghosh. More precisely we have collaborated on model composition for aspect-oriented modelling, model transformation and model validation with testing. Robert France visited Triskell in 2003. Franck Fleurey and Benoit Baudry visited CSU in summer 2005, this visit was funded by INRIA as part of the "mini projet INRIA" program. In january 2006 we started a "Equipe associée" (a three year program for an associated team) called MATT between CSU and Triskell on Model-driven engineering: Aspects, Transformations and Test (see http://www.irisa.fr/triskell/matt for details). Robert France visited Triskell in June 2006 with Sudipto Ghosh and Trung Dinh-Trong. Robert France will also spend 6 months on sabbatical, starting in January 2007. In 2006, Franck Fleurey, Jean-Marie Mottu and Benoit Baudry visited CSU for a month.
- Modelling Simulation and Design Lab, Mc Gill University, Montreal Collaboration on model synthesis and constraint solving issues. Sagar Sen spent 4 months internship in Triskell in Summer 2006.
- Distributed Systems Group, Trinity College, Dublin Collaboration with Andrew Jackson and Siobhan Clarke on validating aspect-oriented models. Andrew Jackson visited Triskell in March 2006 and Jacques Klein, Olivier Barais and Benoit Baudry visited Trinity College in August 2006.
- Carleton University, Ottawa, Canada: Triskell has developed a collaboration on test and objects with Lionel Briand's team at Carleton University.
- Universidad de Murcia, Spain: Cristina Vicente and Diego Caceres have spent the summer 2006 with Triskell to initiate an ongoing collaboration on the use of Kermeta for real-time systems.

# 9. Dissemination

## 9.1. Scientific community animation

### *9.1.1. Journals*

*9.1.1.1. Jean-Marc Jézéquel*

is an Associate Editor of the following journals:

- Journal on Software and System Modeling: SoSyM
- Journal of Object Technology: JOT
- L'Objet

*9.1.1.2. Pierre-Alain Muller*

is an Associate Editor of the following journals:

- Journal on Software and System Modeling: SoSyM

### *9.1.2. Examination Committees*

*9.1.2.1. Jean-Marc Jézéquel*

was in the examination committee of the following PhD thesis and "Habilitation à Diriger les Recherches":

- Ellen Van Paesschen, July 2006, Vrije University Brussels (referee);
- Franck Fleurey, October 2006, université de Rennes (referee);
- Pierre-Alain Muller (HDR), November 2006, université de Rennes (referee);
- Romain Rouvoy, December 2006, université de Lille (referee);
- Wolfgand Theurer, December 2006, Supaero Toulouse (referee);
- Jacques Klein, December 2006, université de Rennes (adviser);
- Jean-Michel Bruel (HDR), December 2006, université de Pau (referee);

*9.1.2.2. Yves Le traon*

was in the examination committee of the following PhD thesis:

- Vu Huy Do, October 2006, ESISAR-INPG (referee);
- Franck Fleurey, October 2006, université de Rennes 1 (adviser);
- Mohammed-Fouz Menai, october 2006, université technologique de Troyes (referee);

*9.1.2.3. Pierre-Alain Muller*

was in the examination committee of the following PhD thesis:

- Franck Fleurey, October 2006, université de Rennes (referee);

*9.1.2.4. Benoit Baudry*

was in the examination committee of the following PhD thesis:

- Franck Fleurey, October 2006, université de Rennes (referee);

### *9.1.3. Conferences*

*9.1.3.1. Jean-Marc Jézéquel*

has been a member of the programme committee of the following conferences:

- MODELS 2006 The 9th International Conference on Model Driven Engineering Languages and Systems (formerly the UML series of conferences) Genoa, Italy, 1-6 October 2006
- CBSE 2006 The 9th International Symposium on Component-Based Software Engineering, Västerås (near Stockholm), Sweden, June 29th -1st July 2006

- SPLC 2006 10th International Software Product Line Conference, Baltimore, Maryland, USA, 21-25 August 2006
- SCESM06 5th Workshop on Scenarios and State Machines: Models, Algorithms and Tools, (associated to ICSE 2006), Shanghai, China, May 2006
- FESCA 2006 Formal Foundations of Embedded Software and Component-Based Software Architectures, Satellite workshop of ETAPS 2006, Sunday 26th March
- Models@run.time Workshop on Models at Run-Time, associated to the 9th International Conference on Model Driven Engineering Languages and Systems, Genoa, Italy, 1-6 October 2006
- 9th International Workshop on Aspect-Oriented Modeling (AOM) October, 2006, Genova, Italy
- QoSA 2006 Second International Conference on the Quality of Software Architectures, Marlardalen University, Sweden, June 27-28, 2006
- LMO 2006, Nimes, March 22-24, 2006
- AFADL'2006, (Approches Formelles dans l'Assistance au Développement de Logiciels), Paris, March 15-17, 2006

#### *9.1.3.2. Yves Le Traon*

has been a member of the programme committee of the following conferences and workshops:

- The 17th IEEE International Symposium on Software Reliability Engineering (ISSRE 2006) November 2006 - Raleigh, North Carolina, USA
- 1st International Conference on Software Data and Technology (ICSOFT 2006).

#### *9.1.3.3. Pierre-Alain Muller*

has been a member of the programme committee of the following conferences:

- MODELS 2006 The 9th International Conference on Model Driven Engineering Languages and Systems (formerly the UML series of conferences) Genoa, Italy, 1-6 October 2006
- International ERCIM Workshop on Software Evolution, Lille, April 2006
- 2èmes journéees sur l'Ingénierie Dirigées par les Modèles (IDM'06), Lille June 2006.

### *9.1.4. Workshops*

J.-M. Jézéquel gave invited talks at:

- Dagstuhl Seminar “Methods for Modelling Software Systems”, Schloss D agstuhl, Germany, August 2006.
- DIPES 2006, 5th IFIP Working Conference on Distributed and Parallel Embedded Systems, Braga, Portugal, October 11-13, 2006.
- Working Meeting on "Models and Aspects for Product Families: A Roadmap for Research", 1st November 2006, Lancaster University.
- Summer School on Aspect Oriented Software Development, Brussels, July 2006.
- Panel on standardization of MDE at MODELS'06, Genoa, October 2006

P.-A. Muller gave invited talks at:

- Journées Neptune 2006: “De la transformation de modèles, application à l'embarqué: la plateforme OpenEmbeDD”, Paris, May 2006.
- “On Metamodels and Language Engineering”, Summer school on Model Driven Development for Distributed Realtime and Embedded Systems, Brest September 06.

Y. Le Traon gave a keynote address at LMO 2006, Nimes, March 22-24, 2006. He was co-organizer of the 2nd workshop on Mutation Analysis, Raleigh, USA.

B. Baudry was co-organizer with Nicolas Rapin (CEA) and Jörn Guy Suess and David Hearnden of the 3rd MoDeVa workshop in conjunction with MoDELS'06. He has also been a member of the programme committee of the following workshops:

- 1st Workshop on Quality in Modeling, (associated to MoDELS'06), Genova, Italy, 1st November 2006.
- 2ème Journée Francophone sur l'Ingénierie Dirigée par les Modèles (IDM'06), Lille, France, 26 - 28 June 2006.

O. Barais was the organizer of the *Second Kermeta day*, which involved 70 peple at Irisa on October 10. He has also been a member of the programme committee of the following workshops:

- 5èmes Journées Composants (JC'06), (associated to RenPar'17 / SympA'2006 / CFSE'5), Perpignan, France, 3-6 october 2006.
- Model Patterns 2006, (associated to IDM'06), Lille, France, 26 - 28 June 2006.

## 9.2. Teaching

Jean-Marc Jézéquel teaches OO Analysis and Design with UML (Iup3 and Diic2) at Ifsic, as well as at Supélec (Rennes) and ENSTB (Rennes). He also gives an advanced course on model driven engineering for Diic3 and MasterPro students.
Noël Plouzeau teaches OO Analysis and Design and Component Based Design to students of the first and second year of Mastère Sciences et Technologies, mention Informatique (Ifsic).
Benoit Baudry teaches software testing (Iup3, Diic3, Master2).
Yves Le Traon teaches V&V for the Master 2 Pro GL.
The Triskell team receives several Master and summer trainees every year.

## 9.3. Miscellaneous

- J.-M. Jézéquel is a member of the Steering Committee of the MODELS/UML Conferences series. He is appointed to the board of the Committee of Projects of INRIA Rennes. He is an expert for the MSTP, DSTP9. He belongs to the evaluation committee of the SIO division of DGA (Direction Générale de l'Armement). He is a member of the scientific council of MIPS (Université de Haute Alsace). He is a Member of the Architecture Board of the MDDi Eclipse project. He participated to the creation of IFIP WG 10.2 on Embedded Systems.
- P.-A. Muller is a member of the Steering Committee of the MODELS/UML Conferences series. He holds an elected position at the scientific council of the Université de Haute Alsace. He wrote the preface for the book “UML pour le développeur”, by Xavier Blanc and Isabelle Mounier, Editions Eyrolles, ISBN : 2-212-12029-X
- J.-M. Jézéquel is the writer of the chapter on Design Patterns in the *Encyclopédie de l'informatique* to be published by Vuibert in the second half of 2006 [17]. He participated to a TV broadcast on France5 (“Emergence d'un nouveau monde”).
- N. Plouzeau is the writer of the chapter on Software Components in the encyclopedia mentioned above [23].

# 10. Bibliography

## Major publications by the team in recent years

[1] B. BAUDRY, F. FLEUREY, J.-M. JÉZÉQUEL, Y. LE TRAON. *Automatic Test Cases Optimization: a Bacteriologic Algorithm*, in "IEEE Software", vol. 22, n$^o$ 2, March 2005, p. 76–82.

[2] A. BEUGNARD, J.-M. JÉZÉQUEL, N. PLOUZEAU, D. WATKINS. *Making Components Contract Aware*, in "IEEE Computer", vol. 13, n$^{o}$ 7, July 1999.

[3] J.-M. JÉZÉQUEL, D. DEVEAUX, Y. LE TRAON. *Reliable Objects: a Lightweight Approach Applied to Java*, in "IEEE Software", vol. 18, n$^{o}$ 4, July/August 2001, p. 76–83.

[4] J.-M. JÉZÉQUEL. *Model Driven Engineering for Distributed Real Time Embedded Systems*, S. GÉRARD, J.-P. BABAU (editors). , chap. Real Time Components and Contracts, Hermes Science Publishing Ltd, London, 2005.

[5] J.-M. JÉZÉQUEL. *Reifying Variants in Configuration Management*, in "ACM Transaction on Software Engineering and Methodology", vol. 8, n$^{o}$ 3, July 1999, p. 284–295.

[6] J.-M. JÉZÉQUEL, M. TRAIN, C. MINGINS. *Design Patterns and Contracts*, ISBN 1-201-30959-9, Addison-Wesley, October 1999.

[7] Y. LE TRAON, B. BAUDRY, J.-M. JÉZÉQUEL. *Design by Contract to improve Software Vigilance*, in "IEEE Transactions on Software Engineering", vol. 32, n$^{o}$ 8, August 2006.

[8] C. NEBUT, F. FLEUREY, Y. LE TRAON, J.-M. JÉZÉQUEL. *Automatic Test Generation: A Use Case Driven Approach*, in "IEEE Trans. on Software Engineering", vol. 32, n$^{o}$ 3, March 2006, p. 140–155.

[9] G. SUNYÉ, A. L. GUENNEC, J.-M. JÉZÉQUEL. *Using UML Action Semantics for Model Execution and Transformation*, in "Information Systems, Elsevier", vol. 27, n$^{o}$ 6, July 2002, p. 445–457.

[10] Y. L. TRAON, T. JÉRON, J.-M. JÉZÉQUEL, P. MOREL. *Efficient OO Integration and Regression Testing*, in "IEEE Trans. on Reliability", vol. 49, n$^{o}$ 1, March 2000, p. 12–25.

[11] T. ZIADI, J.-M. JÉZÉQUEL. *Families Research Book*, LNCS, chap. Product Line Engineering with the UML: Products Derivation, n$^{o}$ to be published, Springer Verlag, 2006.

## Year Publications

### Doctoral dissertations and Habilitation theses

[12] F. FLEUREY. *Langage et méthode pour une ingénierie des modèles fiable*, Ph. D. Thesis, Université de Rennes 1, October 2006.

[13] J. KLEIN. *Aspects Comportementaux et Tissage*, Ph. D. Thesis, Université Rennes 1, 2006.

[14] P.-A. MULLER. *De la modélisation objet des logiciels à la métamodélisation des langages informatiques*, Ph. D. Thesis, Habilitation à diriger les recherches de l'université Rennes 1, 2006.

### Articles in refereed journals and book chapters

[15] J. BAYER, S. GÉRARD, O. HAUGEN, J. MANSELL, B. MOLLER-PEDERSEN, J. OLDEVIK, P. TESSIER, J.-P. THIBAULT, T. WIDEN. *Families Research Book*, LNCS, chap. A Unified Conceptual Model for Product Family Variability Modelling, n$^{o}$ to be published, Springer Verlag, 2006.

[16] J.-M. JÉZÉQUEL, S. GÉRARD, B. BAUDRY. *L'ingénierie dirigée par les modèles*, chap. Le génie logiciel et l'IDM : une approche unificatrice par les modèles, Lavoisier, Hermes-science, 2006.

[17] J.-M. JÉZÉQUEL. *Encyclopédie Vuibert de l'informatique*, chap. Patrons de conception, Vuibert, 2006.

[18] Y. LE TRAON, B. BAUDRY, J.-M. JÉZÉQUEL. *Design by Contract to improve Software Vigilance*, in "IEEE Transactions on Software Engineering", vol. 32, n$^{o}$ 8, August 2006, p. 571–586.

[19] P.-A. MULLER. *From MDD Concepts to Experiments and Illustrations*, chap. On Metamodels and Language Engineering, ISTE, ISBN 1905209592, 2006.

[20] C. NEBUT, F. FLEUREY, Y. LE TRAON, J.-M. JÉZÉQUEL. *Automatic Test Generation: A Use Case Driven Approach*, in "IEEE Transactions on Software Engineering", vol. 32, n$^{o}$ 3, March 2006, p. 140–155.

[21] C. NEBUT, Y. LE TRAON, J.-M. JÉZÉQUEL. *Families Research Book*, LNCS, chap. System Testing of Product Families: from Requirements to Test Cases, n$^{o}$ to be published, Springer Verlag, 2006.

[22] S. PICKIN, C. JARD, T. JÉRON, J.-M. JÉZÉQUEL, Y. LE TRAON. *Test Synthesis from UML Models of Distributed Software*, in "IEEE Transactions on Software Engineering", vol. 32, n$^{o}$ 12, December 2006.

[23] N. PLOUZEAU. *Encyclopédie Vuibert de l'informatique*, chap. Composants logiciels, Vuibert, 2006.

[24] J. STEEL, J.-M. JÉZÉQUEL. *On Model Typing*, in "Journal of Software and Systems Modeling (SoSyM)", 2006.

[25] T. ZIADI, J.-M. JÉZÉQUEL. *Families Research Book*, LNCS, chap. Product Line Engineering with the UML: Products Derivation, n$^{o}$ to be published, Springer Verlag, 2006.

## Publications in Conferences and Workshops

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

[27] B. BAUDRY, T. DINH-TRONG, J.-M. MOTTU, D. SIMMONDS, R. FRANCE, S. GHOSH, F. FLEUREY, Y. LE TRAON. *Model Transformation Testing Challenges*, in "ECMDA workshop on Integration of Model Driven Development and Model Driven Testing., Bilbao, Spain", July 2006.

[28] B. BAUDRY, F. FLEUREY, Y. LE TRAON. *Improving Test Suites for Efficient Fault Localization*, in "28th International Conference on Software Engineering (ICSE 06)", selection : 9%, ACM, 2006.

[29] E. BROTTIER, F. FLEUREY, J. STEEL, B. BAUDRY, Y. LE TRAON. *Metamodel-based Test Generation for Model Transformations: an Algorithm and a Tool*, in "Proceedings of ISSRE'06, Raleigh, NC, USA", November 2006.

[30] W. CAZZOLA, J.-M. JÉZÉQUEL, A. RASHID. *Semantic Join Point Models: Motivations, Notions and Requirements*, in "SPLAT 2006 (Software Engineering Properties of Languages and Aspect Technologies)", March 2006.

[31] R. DELAMARE, B. BAUDRY, Y. LE TRAON. *Reverse-engineering of UML 2.0 Sequence Diagrams from Execution Traces*, in "Workshop on Object-Oriented Reengineering at ECOOP 06, Nantes, France", July 2006.

[32] A. JACKSON, O. BARAIS, J.-M. JÉZÉQUEL, S. CLARKE. *Toward A Generic And Extensible Merge*, in "Models and Aspects workshop, at ECOOP 2006, Nantes, France", July 2006.

[33] A. JACKSON, J. KLEIN, B. BAUDRY, S. CLARKE. *KerTheme: Testing Aspect Oriented Models*, in "ECMDA workshop on Integration of Model Driven Development and Model Driven Testing., Bilbao, Spain", July 2006.

[34] A. JACKSON, J. KLEIN, B. BAUDRY, S. CLARKE. *Testing Executable Themes*, in "In Second Workshop on Models and Aspects, Handling Crosscutting Concerns in MDSD at ECOOP 06, Nantes, France", July 2006.

[35] J.-M. JÉZÉQUEL. *Model Driven Aspect Weaving*, in "1st European Summer School on Aspect-oriented Software Development, Vrije Universiteit Brussel, Belgium", July 2006.

[36] J.-M. JÉZÉQUEL. *Reifying the Semantic Domains of Component Contracts*, in "5th IFIP Working Conference on Distributed and Parallel Embedded Systems, DIPES'06, Braga, Portugal", Springer SBM, October 2006.

[37] J. KLEIN, F. FLEUREY. *Tissage d'Aspects Comportementaux*, in "Langages et Modèles à Objets: LMO'06, Nimes, France", March 2006.

[38] J. KLEIN, L. HÉLOUET, J.-M. JÉZÉQUEL. *Semantic-based Weaving of Scenarios*, in "proceedings of the 5th International Conference on Aspect-Oriented Software Development (AOSD'06), Bonn, Germany", ACM, March 2006.

[39] J.-M. MOTTU, B. BAUDRY, Y. LE TRAON. *Mutation Analysis Testing for Model Transformations*, in "proceedings of the European Conference on Model Driven Architecture (ECMDA 06), Bilbao, Spain", July 2006.

[40] J.-M. MOTTU, B. BAUDRY, Y. LE TRAON. *Reusable MDA Components: A Testing-for-Trust Approach*, in "proceedings of the MoDELS/UML 2006, Genova, Italy", October 2006.

[41] P.-A. MULLER, F. FLEUREY, F. FONDEMENT, M. HASSENFORDER, R. SCHNECKENBURGER, S. GÉRARD, J.-M. JÉZÉQUEL. *Model-Driven Analysis and Synthesis of Concrete Syntax*, in "Proceedings of the MoDELS/UML 2006, Genova, Italy", October 2006.

[42] C. NEBUT, B. BAUDRY, S. KAMOUN, W. AHMED SAEED. *Multi-Language Support for Model-Driven Requirement Analysis and Test Generation*, in "ECMDA workshop on Integration of Model Driven Development and Model Driven Testing., Bilbao, Spain", July 2006.

[43] S. SAUDRAIS, O. BARAIS, L. DUCHIEN. *Using Model-Driven Engineering to generate QoS Monitors from a formal specification*, in "Proceedings of the Aquserm 2006, Hong Kong, China", October 2006.

[44] S. SAUDRAIS, O. BARAIS, N. PLOUZEAU. *Composants avec Propriétés Temporelles*, in "Proceedings of the CAL 2006, Nantes, France", September 2006.

[45] S. SEN, B. BAUDRY. *Mutation-based Model Synthesis in Model Driven Engineering*, in "Mutation'06 workshop associtaed to ISSRE'06, Raleigh, NC, USA", November 2006.

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

### Internal Reports

[47] Z. DREY, C. FAUCHER, F. FLEUREY, D. VOJTISEK. *Kermeta language reference manual*, 2006, http://www.kermeta.org/documents/manual/.

[48] Z. DREY, D. VOJTISEK. *Kermeta EMF tutorial*, 2006, http://www.kermeta.org/documents/Tutorial/.

[49] D. VOJTISEK. *Kermeta user interface guide*, 2006, http://www.kermeta.org/documents/ui_user_guide/.

### Miscellaneous

[50] O. BARAIS. *SpoonEMF, une brique logicielle pour l'utilisation de l'IDM dans le cadre de la réingénierie de programmes Java5*, June 2006, http://planetmde.org/idm06/posters/11.pdf, 2 ème Journée sur l'Ingénierie Dirigée par les Modèles.

[51] J.-M. JÉZÉQUEL. *L'ingénierie des modèles*, October 2006, Performance, Silicomp-AQL.

[52] D. VOJTISEK. *QVT : un standard de transformation pour l'Ingénierie Dirigée par les Modèles*, http://www.standarmedia.com, January 2006, http://www.standarmedia.com/std/acc_det.asp?Ids=1A32&Ref=564&Page=1&lang

## References in notes

[53] A. BEUGNARD, J.-M. JÉZÉQUEL, N. PLOUZEAU, D. WATKINS. *Making Components Contract Aware*, in "IEEE Computer", vol. 13, n$^o$ 7, July 1999.

[54] G. BOOCH. *Object-Oriented Analysis and Design with Applications*, 2nd, Benjamin Cummings, 1994.

[55] L. CASTELLANO, G. DE MICHELIS, POMELLO,L.. *Concurrency versus Interleaving: An Instructive Example*, in "BEATCS: Bulletin of the European Association for Theoretical Computer Science", vol. 31, 1987.

[56] E. GAMMA, R. HELM, R. JOHNSON, J. VLISSIDES. *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison Wesley, 1995.

[57] M. JACKSON. *System Development*, Prentice-Hall International, Series in Computer Science, 1985.

[58] J.-M. JÉZÉQUEL, B. MEYER. *Design by Contract: The Lessons of Ariane*, in "Computer", vol. 30, n$^o$ 1, January 1997, p. 129–130.

[59] B. MEYER. *Reusability: The Case for Object-Oriented Design*, in "IEEE SOFTWARE", n$^o$ 3, March 1987, p. 50–64.

[60] B. MEYER. *Applying "Design by Contract"*, in "IEEE Computer (Special Issue on Inheritance & Classification)", vol. 25, n$^o$ 10, October 1992, p. 40–52.

[61] A. RASHID, J. ARAÚJO. *Modularisation and composition of aspectual requirements*, in "Proceedings of the 2nd international conference on Aspect-oriented software development", 2003, p. 11–20.

[62] C. SZYPERSKI. *Component Software: Beyond Object-Oriented Programming*, ACM Press and Addison-Wesley, New York, N.Y., 1998.

[63] J. WARMER, A. KLEPPE. *The Object Constraint Language*, Addison-Wesley, 1998.

[64] G. WINSKEL. *Event Structures*, in "Petri Nets: Applications and Relationships to Other Models of Concurrency, Advances in Petri Nets 1986, Part II, Proceedings of an Advanced Course, Bad Honnef", W. BRAUER, W. REISIG, G. ROZENBERG (editors). , vol. 255, Springer-Verlag, september 1986, p. 325-392.