
IN PARTNERSHIP WITH:
**Université des sciences et technologies de Lille (Lille 1)**

Activity Report 2015

# Project-Team RMOD

## Analyses and Languages Constructs for Object-Oriented Application Evolution

IN COLLABORATION WITH: Centre de Recherche en Informatique, Signal et Automatique de Lille

RESEARCH CENTER
**Lille - Nord Europe**

THEME
**Distributed programming and software engineering**

# Table of contents

## Project-Team RMOD

*Creation of the Project-Team: 2009 July 01*

**Keywords:**

**Computer Science and Digital Science:**

2. - Software
2.1. - Programming Languages
2.1.10. - Domain-specific languages
2.1.2. - Object-oriented programming
2.1.4. - Aspect-oriented programming
2.1.9. - Dynamic languages
2.5. - Software engineering
2.6. - Infrastructure software
2.6.3. - Virtual machines

**Other Research Topics and Application Domains:**

2. - Health
2.7. - Medical devices
5. - Industry of the future
5.9. - Industrial maintenance
6.5. - Information systems
7. - Transport and logistics

# 1. Members

**Research Scientists**

Stéphane Ducasse [Team leader, Inria, Senior Researcher, HdR]
Marcus Denker [Inria, Researcher]

**Faculty Members**

Nicolas Anquetil [Univ. Lille I, Associate Professor, HdR]
Jean-Christophe Bach [Telecom Bretagne, Associate Professor]
Damien Cassou [Univ. Lille I, Associate Professor]
Anne Etien [Univ. Lille I, Associate Professor]
Damien Pollet [Univ. Lille I, Associate Professor]

**Engineers**

Jean-Baptiste Arnaud [Inria, until Jun 2015, granted by OSEO Innovation]
Denis Kudriashov [Inria, from Nov 2015]
Esteban Lorenzano [Inria, granted by OSEO Innovation]
Guillermo Polito [Inria, until Aug 2015]
Olivier Auverlot [Univ. Lille I, Engineer 30%]
Christophe Demarey [Inria, Engineer 70%]

**PhD Students**

Clément Béra [Inria, granted by Région Nord-Pas-de-Calais]
Vincent Blondeau [Worldline, granted by CIFRE]
Gustavo Jansen de Souza Santos [Lille, granted by CNPq Brasil]

Victor Martin Dias [Inria, until Nov 2015, granted by Région Nord-Pas-de-Calais]
Brice Govin [Thales, granted by CIFRE]
Max Mattone [Ecole des Mines de Douai, until Oct 2015]
Marco Naddeo [University of Turin, co-supervision, from Jan 2014]
Camille Teruel [Inria, until Nov 2015]
Thibault Raffaillac [Inria, from Nov 2015, collaboration with Mjolnir]
Pablo Tesone [Inria, from Aug 2015]
Klerisson Vinicius Ribeiro Da Paixao [UFU - Federal University of Uberlândia, Brazil, from Sep 2015]

**Post-Doctoral Fellows**

Stefan Marr [Inria, until Apr 2015]
Ricardo Terra Nunes Bueno Villela [Federal university of Lavras (Brazil) , until Mar 2015]

**Visiting Scientists**

Abdelghani Alidra [Associate Professor, University 20 Aout 55, Skikda. Algeria, from May 2015]
Alexandre Bergel [University of Chile, Researcher, until Jul 2015]
Miguel Campusano Araya [University of Chile, until Oct 2015]
Glenn Cavarle [CNRS, Jun 2015]
Guido Chari [UBA Argentina, until Mar 2015]
Leonardo Guimaraes Silva [Federal University of Minas Gerais (Brazil), until Jun 2015]
Matthieu Lacaton [Thales, Apr 2015]
Alain Plantec [Univ. Bretagne Occidentale, until Jun 2015]
Markiyan Rizun [Ivan Franko National University of Lviv, Ukraine, until Aug 2015]

**Administrative Assistant**

Julie Jonas [Inria]

**Others**

Pierre Chanson [ObjectProfile, Chile, Jul 2015]
Sebastien Besse [Inria, Intern, until Jun 2015]
Cyril Ferlicot-Delbecque [Univ. Lille I, Intern, from Mar 2015 until Aug 2015]
Thomas Heniart [Inria, Intern, from Mar 2015 until Aug 2015]
Kevin Lanvin [Inria, Intern, until Jul 2015]
Skip Lentz [Inria, Intern, from Aug 2015]
Merwan Ouddane [Univ. Lille I, Intern, from Mar 2015 until Aug 2015]
Franck Warlouzet [Univ. Lille I, Intern, from Mar 2015 until Aug 2015]

# 2. Overall Objectives

## 2.1. Introduction

**Keywords:** Software evolution, Maintenance, Program visualization, Program analyses, Meta modelling, Software metrics, Quality models, Object-oriented programming, Reflective programming, Traits, Dynamically typed languages, Smalltalk.

RMoD's general vision is defined in two objectives: remodularization and modularity constructs. These two views are the two faces of a same coin: maintenance could be eased with better engineering and analysis tools and programming language constructs could let programmers define more modular applications.

## 2.2. Reengineering and remodularization

While applications must evolve to meet new requirements, few approaches analyze the implications of their original structure (modules, packages, classes) and their transformation to support their evolution. Our research will focus on the *remodularization* of object-oriented applications. Automated approaches including clustering algorithms are not satisfactory because they often ignore user inputs. Our vision is that we need better approaches to support the transformation of existing software. The reengineering challenge tackled by RMoD is formulated as follows:

*How to help remodularize existing software applications?*

We are developing analyses and algorithms to remodularize object-oriented applications. This is why we started studying and building tools to support the *understanding of applications* at the level of packages and modules. This allows us to understand the results of the *analyses* that we are building.

## 2.3. Constructs for modular and isolating programming languages

Dynamically-typed programming languages such as JavaScript are getting new attention as illustrated by the large investment of Google in the development of the Chrome V8 JavaScript engine and the development of a new dynamic language DART. This new trend is correlated to the increased adoption of dynamic programming languages for web-application development, as illustrated by Ruby on Rails, PHP and JavaScript. With web applications, users expect applications to be always available and getting updated on the fly. This continuous evolution of application is a real challenge [67]. Hot software evolution often requires *reflective* behavior and features. For instance in CLOS and Smalltalk each class modification automatically migrates existing instances on the fly.

At the same time, there is a need for *software isolation i.e.,*, applications should reliably run co-located with other applications in the same virtual machine with neither confidential information leaks nor vulnerabilities. Indeed, often for economical reasons, web servers run multiple applications on the same virtual machine. Users need confined applications. It is important that (1) an application does not access information of other applications running on the same virtual machine and (2) an application authorized to manipulate data cannot pass such authorization or information to other parts of the application that should not get access to it.

Static analysis tools have always been confronted to reflection [64]. Without a full treatment of reflection, static analysis tools are both incomplete and unsound. Incomplete because some parts of the program may not be included in the application call graph, and unsound because the static analysis does not take into account reflective features [73]. In reflective languages such as F-Script, Ruby, Python, Lua, JavaScript, Smalltalk and Java (to a certain extent), it is possible to nearly change any aspect of an application: change objects, change classes dynamically, migrate instances, and even load untrusted code.

Reflection and isolation concerns are a priori antagonistic, pulling language design in two opposite directions. Isolation, on the one hand, pulls towards more static elements and types (*e.g.,*, ownership types). Reflection, on the other hand, pulls towards fully dynamic behavior. This tension is what makes this a real challenge: As experts in reflective programming, dynamic languages and modular systems, we believe that by working on this important tension we can make a breakthrough and propose innovative solutions in resolving or mitigating this tension. With this endeavor, we believe that we are working on a key challenge that can have an impact on future programming languages. The language construct challenge tackled by RMoD is formulated as follows:

*What are the language modularity constructs to support isolation?*

In parallel we are continuing our research effort on traits [1] by assessing trait scalability and reuse on a large case study and developing a pure trait-based language. In addition, we dedicate efforts to remodularizing a meta-level architecture in the context of the design of an isolating dynamic language. Indeed at the extreme, modules and structural control of reflective features are the first steps towards flexible, dynamic, yet isolating, languages. As a result, we expect to demonstrate that having adequate composable units and scoping units will help the evolution and recomposition of an application.

# 3. Research Program

## 3.1. Software Reengineering

Strong coupling among the parts of an application severely hampers its evolution. Therefore, it is crucial to answer the following questions: How to support the substitution of certain parts while limiting the impact on others? How to identify reusable parts? How to modularize an object-oriented application?

[1] Traits are groups of methods that can be composed orthogonally to simple inheritance. Contrary to mixin, the class has the control of the composition and conflict management.

Having good classes does not imply a good application layering, absence of cycles between packages and reuse of well-identified parts. Which notion of cohesion makes sense in presence of late-binding and programming frameworks? Indeed, frameworks define a context that can be extended by subclassing or composition: in this case, packages can have a low cohesion without being a problem for evolution. How to obtain algorithms that can be used on real cases? Which criteria should be selected for a given remodularization?

To help us answer these questions, we work on enriching Moose, our reengineering environment, with a new set of analyses [58], [57]. We decompose our approach in three main and potentially overlapping steps:

1. Tools for understanding applications,
2. Remodularization analyses,
3. Software Quality.

### 3.1.1. Tools for understanding applications

**Context and Problems.** We are studying the problems raised by the understanding of applications at a larger level of granularity such as packages or modules. We want to develop a set of conceptual tools to support this understanding.

Some approaches based on Formal Concept Analysis (FCA) [86] show that such an analysis can be used to identify modules. However the presented examples are too small and not representative of real code.

**Research Agenda.**

FCA provides an important approach in software reengineering for software understanding, design anomalies detection and correction, but it suffers from two problems: (i) it produces lattices that must be interpreted by the user according to his/her understanding of the technique and different elements of the graph; and, (ii) the lattice can rapidly become so big that one is overwhelmed by the mass of information and possibilities [47]. We look for solutions to help people putting FCA to real use.

### 3.1.2. Remodularization analyses

**Context and Problems.** It is a well-known practice to layer applications with bottom layers being more stable than top layers [74]. Until now, few works have attempted to identify layers in practice: Mudpie [88] is a first cut at identifying cycles between packages as well as package groups potentially representing layers. DSM (dependency structure matrix) [87], [82] seems to be adapted for such a task but there is no serious empirical experience that validates this claim. From the side of remodularization algorithms, many were defined for procedural languages [70]. However, object-oriented programming languages bring some specific problems linked with late-binding and the fact that a package does not have to be systematically cohesive since it can be an extension of another one [89], [61].

As we are designing and evaluating algorithms and analyses to remodularize applications, we also need a way to understand and assess the results we are obtaining.

**Research Agenda.** We work on the following items:

Layer identification. We propose an approach to identify layers based on a semi-automatic classification of package and class interrelationships that they contain. However, taking into account the wish or knowledge of the designer or maintainer should be supported.

Cohesion Metric Assessment. We are building a validation framework for cohesion/coupling metrics to determine whether they actually measure what they promise to. We are also compiling a number of traditional metrics for cohesion and coupling quality metrics to evaluate their relevance in a software quality setting.

### 3.1.3. Software Quality

**Research Agenda.** Since software quality is fuzzy by definition and a lot of parameters should be taken into account we consider that defining precisely a unique notion of software quality is definitively a Grail in the realm of software engineering. The question is still relevant and important. We work on the two following items:

Quality models. We studied existing quality models and the different options to combine indicators — often, software quality models happily combine metrics, but at the price of losing the explicit relationships between the indicator contributions. There is a need to combine the results of one metric over all the software components of a system, and there is also the need to combine different metric results for any software component. Different combination methods are possible that can give very different results. It is therefore important to understand the characteristics of each method.

Bug prevention. Another aspect of software quality is validating or monitoring the source code to avoid the emergence of well known sources of errors and bugs. We work on how to best identify such common errors, by trying to identify earlier markers of possible errors, or by helping identifying common errors that programmers did in the past.

## 3.2. Language Constructs for Modular Design

While the previous axis focuses on how to help remodularizing existing software, this second research axis aims at providing new language constructs to build more flexible and recomposable software. We will build on our work on traits [84], [59] and classboxes [48] but also start to work on new areas such as isolation in dynamic languages. We will work on the following points: (1) Traits and (2) Modularization as a support for isolation.

### *3.2.1. Traits-based program reuse*

**Context and Problems.** Inheritance is well-known and accepted as a mechanism for reuse in object-oriented languages. Unfortunately, due to the coarse granularity of inheritance, it may be difficult to decompose an application into an optimal class hierarchy that maximizes software reuse. Existing schemes based on single inheritance, multiple inheritance, or mixins, all pose numerous problems for reuse.

To overcome these problems, we designed a new composition mechanism called Traits [84], [59]. Traits are pure units of behavior that can be composed to form classes or other traits. The trait composition mechanism is an alternative to multiple or mixin inheritance in which the composer has full control over the trait composition. The result enables more reuse than single inheritance without introducing the drawbacks of multiple or mixin inheritance. Several extensions of the model have been proposed [56], [78], [49], [60] and several type systems were defined [62], [85], [79], [72].

Traits are reusable building blocks that can be explicitly composed to share methods across unrelated class hierarchies. In their original form, traits do not contain state and cannot express visibility control for methods. Two extensions, stateful traits and freezable traits, have been proposed to overcome these limitations. However, these extensions are complex both to use for software developers and to implement for language designers.

**Research Agenda: Towards a pure trait language.** We plan distinct actions: (1) a large application of traits, (2) assessment of the existing trait models and (3) bootstrapping a pure trait language.

- To evaluate the expressiveness of traits, some hierarchies were refactored, showing code reuse [51]. However, such large refactorings, while valuable, may not exhibit all possible composition problems, since the hierarchies were previously expressed using single inheritance and following certain patterns. We want to redesign from scratch the collection library of Smalltalk (or part of it). Such a redesign should on the one hand demonstrate the added value of traits on a real large and redesigned library and on the other hand foster new ideas for the bootstrapping of a pure trait-based language.

  In particular we want to reconsider the different models proposed (stateless [59], stateful [50], and freezable [60]) and their operators. We will compare these models by (1) implementing a trait-based collection hierarchy, (2) analyzing several existing applications that exhibit the need for traits. Traits may be flattened [77]. This is a fundamental property that confers to traits their simplicity and expressiveness over Eiffel's multiple inheritance. Keeping these aspects is one of our priority in forthcoming enhancements of traits.

- Alternative trait models. This work revisits the problem of adding state and visibility control to traits. Rather than extending the original trait model with additional operations, we use a fundamentally different approach by allowing traits to be lexically nested within other modules. This enables traits to express (shared) state and visibility control by hiding variables or methods in their lexical scope. Although the traits' "flattening property" no longer holds when they can be lexically nested, the combination of traits with lexical nesting results in a simple and more expressive trait model. We formally specify the operational semantics of this combination. Lexically nested traits are fully implemented in AmbientTalk, where they are used among others in the development of a Morphic-like UI framework.
- We want to evaluate how inheritance can be replaced by traits to form a new object model. For this purpose we will design a minimal reflective kernel, inspired first from ObjVlisp [55] then from Smalltalk [65].

### *3.2.2. Reconciling Dynamic Languages and Isolation*

**Context and Problems.** More and more applications require dynamic behavior such as modification of their own execution (often implemented using reflective features [69]). For example, F-script allows one to script Cocoa Mac-OS X applications and Lua is used in Adobe Photoshop. Now in addition more and more applications are updated on the fly, potentially loading untrusted or broken code, which may be problematic for the system if the application is not properly isolated. Bytecode checking and static code analysis are used to enable isolation, but such approaches do not really work in presence of dynamic languages and reflective features. Therefore there is a tension between the need for flexibility and isolation.

**Research Agenda: Isolation in dynamic and reflective languages.** To solve this tension, we will work on *Sure*, a language where isolation is provided by construction: as an example, if the language does not offer field access and its reflective facilities are controlled, then the possibility to access and modify private data is controlled. In this context, layering and modularizing the meta-level [52], as well as controlling the access to reflective features [53], [54] are important challenges. We plan to:

- Study the isolation abstractions available in erights (http://www.erights.org) [76], [75], and Java's class loader strategies [71], [66].
- Categorize the different reflective features of languages such as CLOS [68], Python and Smalltalk [80] and identify suitable isolation mechanisms and infrastructure [63].
- Assess different isolation models (access rights, capabilities [81]...) and identify the ones adapted to our context as well as different access and right propagation.
- Define a language based on
  - the decomposition and restructuring of the reflective features [52],
  - the use of encapsulation policies as a basis to restrict the interfaces of the controlled objects [83],
  - the definition of method modifiers to support controlling encapsulation in the context of dynamic languages.

An open question is whether, instead of providing restricted interfaces, we could use traits to grant additional behavior to specific instances: without trait application, the instances would only exhibit default public behavior, but with additional traits applied, the instances would get extra behavior. We will develop *Sure*, a modular extension of the reflective kernel of Smalltalk (since it is one of the languages offering the largest set of reflective features such as pointer swapping, class changing, class definition...) [80].

# 4. Application Domains

## 4.1. Programming Languages and Tools

Many of the results of RMoD are improving programming languages or development tools for such languages. As such the application domain of these results is as varied as the use of programming languages in general. Pharo, the language that RMoD develops, is used for a very broad range of applications. From pure research experiments to real world industrial use (the Pharo Consortium has around 20 company members) http://consortium.pharo.org Examples are web applications, server backends for mobile applications or even graphical tools and embedded applications.

## 4.2. Software Reengineering

Moose is a language-independent environment for reverse- and re-engineering complex software systems. Moose provides a set of services including a common meta-model, metrics evaluation and visualization. As such Moose is used for analysing software systems to support understanding and continous development as well as software quality analysis.

# 5. Highlights of the Year

## 5.1. Highlights of the Year

- Pharo 4.0 has been released in April 2015.
- Moose 5.1 has been released in June 2015.
- The Synectique company, a spin-off of the RMod group with two members actively participating, got selected on the i-Lab 2015 contest (category: Creation and Development). 364 projects were submitted in this category and 54 got selected ($<15\%$). This will allow the young company to expand its activities by hiring young developers and a sales person.
- Papers published at PLDI and OOPSLA, two important conferences of our field.

### *5.1.1. Awards*

- The paper : **First Analysis of String APIs: the Case of Pharo [36] got a price at IWST 15 International Workshop On Smalltalk Technologies.**
- **A paper of Martin Dias [25] was a candidate for best paper (part of the best 5) at SANER http://saner.soccerlab.polymtl.ca/doku.php?id=en:awards**
- **Markiyan Rizun got the third price at ESUG 2015 for his Rewrite tool.**

# 6. New Software and Platforms

## 6.1. Moose

FUNCTIONAL DESCRIPTION

Moose is an extensive platform for software and data analysis. It offers multiple services ranging from importing and parsing data, to modeling, to measuring, querying, mining, and to building interactive and visual analysis tools.

- Participants: Stéphane Ducasse, Muhammad Bhatti, Andre Cavalcante Hora, Nicolas Anquetil, Anne Etien, Guillaume Larcheveque and Alexandre Bergel
- Partners: Université de Berne - Sensus - Synectique - Pleiad - USI - Vrije Universiteit Brussel
- Contact: Stéphane Ducasse
- URL: http://www.moosetechnology.org

## 6.2. Pharo

KEYWORDS: Live programmation objet - Reflective system
FUNCTIONAL DESCRIPTION

The platform Pharo is an open-source Smalltalk-inspired language and environment. It provides a platform for innovative development both in industry and research. By providing a stable and small core system, excellent developer tools, and maintained releases, Pharo's goal is to be a platform to build and deploy mission critical applications, while at the same time continue to evolve.

- Participants: Marcus Denker, Damien Cassou, Christophe Demarey, Stéphane Ducasse, Esteban Lorenzano, Damien Pollet, Camille Teruel and Clément Béra
- Partners: Université de Berne - École des Mines de Douai - Uqbar foundation Argentina - Sensus - Synectique - Pleiad - Debris publishing - Yesplan - HR Works - MAD - BetaNine - Vmware
- Contact: Marcus Denker
- URL: http://www.pharo.org

## 6.3. Pillar

KEYWORDS: HTML - LaTeX - HTML5
FUNCTIONAL DESCRIPTION

Pillar is a markup syntax and associated tools to write and generate documentation and books. Pillar is currently used to write several books and other documentation. Two platforms have already been created on top of Pillar: PillarHub and Marina.

- Contact: Damien Cassou
- URL: http://www.smalltalkhub.com/#!/~Pier/Pillar

# 7. New Results

## 7.1. Tools for understanding evolution

**Automatic Detection of System-Specific Conventions.** In Apache Ant, a convention to improve maintenance was introduced in 2004 stating a new way to close files instead of the Java generic InputStream.close(). Yet, six years after its introduction, this convention was still not generally known to the developers. Two existing solutions could help in these cases. First, one can deprecate entities, but, in our example, one can hardly deprecate Java's method. Second, one can create a system-specific rule to be automatically enforced. In a preceding publication , we showed that system-specific rules are more likely to be noticed by developers than generic ones. However, in practice, developers rarely create specific rules. We therefore propose to free the developers from the need to create rules by automatically detecting such conventions from source code repositories. This is done by mining the change history of the system to discover similar changes being applied over several revisions. The proposed approach is applied to real-world systems, and the extracted rules are validated with the help of experts. The results show that many rules are in fact relevant for the experts. [16]

**DeltaImpactFinder.** In software development, version control systems (VCS) provide branching and merging support tools. Such tools are popular among developers to concurrently change a code-base in separate lines and reconcile their changes automatically afterwards. However, two changes that are correct independently can introduce bugs when merged together. We call semantic merge conflicts this kind of bugs. Change impact analysis (CIA) aims at estimating the effects of a change in a codebase. We propose to detect semantic merge conflicts using CIA. On a merge, DELTAIMPACTFINDER analyzes and compares the impact of a change in its origin and destination branches. We call the difference between these two impacts the delta-impact. If the delta-impact is empty, then there is no indicator of a semantic merge conflict and the merge can continue automatically. Otherwise, the delta-impact contains what are the sources of possible conflicts. [26]

**OrionPlanning.** Many techniques have been proposed in the literature to support architecture definition, conformance, and analysis. However, there is a lack of adoption of such techniques by the industry. Previous work have analyzed this poor support. Specifically, former approaches lack proper analysis techniques (e.g., detection of architectural inconsistencies), and they do not provide extension and addition of new features. We present ORIONPLANNING, a prototype tool to assist refactorings at large scale. The tool provides support for model-based refactoring operations. These operations are performed in an interactive visualization. The contributions of the tool consist in: (i) providing iterative modifications in the architecture, and (ii) providing an environment for architecture inspection and definition of dependency rules. [37]

**Recording and Replaying System-Specific Conventions.** During its lifetime, a software system is under continuous maintenance to remain useful. Maintenance can be achieved in activities such as adding new features, fixing bugs, improving the system's structure, or adapting to new APIs. In such cases, developers sometimes perform sequences of code changes in a systematic way. These sequences consist of small code changes (e.g., create a class, then extract a method to this class), which are applied to groups of related code entities (e.g., some of the methods of a class). MacroRecorder is a proof-of-concept tool that records a sequence of code changes, then it allows the developer to generalize this sequence in order to apply it in other code locations. The evaluation is based on previous work on repetitive code changes related to rearchitecting. MacroRecorder was able to replay 92% of the examples, which consisted in up to seven code entities modified up to 66 times. The generation of a customizable, large-scale transformation operator has the potential to efficiently assist code maintenance. [39], [38]

## 7.2. Software Quality: Taming Software Evolution

**Software metrics do not predict the health of a project.** More and more companies would like to mine software data with the goal of assessing the health of their software projects. The hope is that some software metrics could be tracked to predict failure risks or confirm good health. If a factor of success was found, projects failures could be anticipated and early actions could be taken by the organisation to help or to monitor closely the project, allowing one to act in a preventive mode rather than a curative one. We were called by a major IT company to fulfil this goal. We conducted a study to check whether software metrics can be related to project failure. The study was both theoretic with a review of literature on the subject, and practical with mining past projects data and interviews with project managers. We found that metrics used in practice are not reliable to assess project outcome. [22]

**How Do Developers React to API Evolution?** Software engineering research now considers that no system is an island, but it is part of an ecosystem involving other systems, developers, users, hardware,. .. When one system (e.g., a framework) evolves, its clients often need to adapt. Client developers might need to adapt to functionalities, client systems might need to be adapted to a new API, client users might need to adapt to a new User Interface. The consequences of such changes are yet unclear, what proportion of the ecosystem might be expected to react, how long might it take for a change to diffuse in the ecosystem, do all clients react in the same way? We report on an exploratory study aimed at observing API evolution and its impact on a large-scale software ecosystem, Pharo, which has about 3,600 distinct systems, more than 2,800 contributors, and six years of evolution. We analyze 118 API changes and answer research questions regarding the magnitude, duration, extension, and consistency of such changes in the ecosystem. The results of this study help to characterize the impact of API evolution in large software ecosystems, and provide the basis to better understand how such impact can be alleviated. [27]

**Does JavaScript software embrace classes?** JavaScript is the de facto programming language for the Web. It is used to implement mail clients, office applications, or IDEs, that can weight hundreds of thousands of lines of code. The language itself is prototype based, but to master the complexity of their application, practitioners commonly rely on some informal class abstractions. This practice has never been the target of empirical investigations in JavaScript. Yet, understanding it would be key to adequately tune programming environments and structure libraries such as they are accessible to programmers. We report a large and in-depth study to understand how class emulation is employed in JavaScript applications. We propose a strategy to statically detect class-based abstractions in the source code of JavaScript systems. We used this strategy in

a dataset of 50 popular JavaScript applications available from GitHub. We found systems structured around hundreds of classes, suggesting that JavaScript developers are standing on traditional class-based abstractions to tackle the growing complexity of their systems. [28]

## 7.3. Software Quality: History and Changes

**Mining Architectural Violations from Version History.** Software architecture conformance is a key software quality control activity that aims to reveal the progressive gap normally observed between concrete and planned software architectures. However, formally specifying an architecture can be difficult, as it must be done by an expert of the system having a high level understanding of it. We present a lightweighted approach for architecture conformance based on a combination of static and historical source code analysis. The proposed approach relies on four heuristics for detecting absences (something expected was not found) and divergences (something prohibited was found) in source code based architectures. We also present an architecture conformance process based on the proposed approach. We followed this process to evaluate the architecture of two industrial-strength information systems, achieving an overall precision of 62.7% and 53.8%. We also evaluated our approach in an open-source information retrieval library, achieving an overall precision of 59.2%. We envision that an heuristic-based approach for architecture conformance can be used to rapidly raise architectural warnings, without deeply involving experts in the process. [17]

**Untangling Fine-Grained Code Changes.** After working for some time, developers commit their code changes to a version control system. When doing so, they often bundle unrelated changes (e.g., bug fix and refactoring) in a single commit, thus creating a so-called tangled commit. Sharing tangled commits is problematic because it makes review, reversion, and integration of these commits harder and historical analyses of the project less reliable. Researchers have worked at untangling existing commits, i.e., finding which part of a commit relates to which task. We contribute to this line of work in two ways: (1) A publicly available dataset of untangled code changes, created with the help of two developers who accurately split their code changes into self contained tasks over a period of four months; (2) a novel approach, EpiceaUntangler, to help developers share untangled commits (aka. atomic commits) by using fine-grained code change information. EpiceaUntangler is based and tested on the publicly available dataset, and further evaluated by deploying it to 7 developers, who used it for 2 weeks. We recorded a median success rate of 91% and average one of 75%, in automatically creating clusters of untangled fine-grained code changes. [25]

**Developers' Perception of Co-Change Patterns: An Empirical Study**. Co-change clusters are groups of classes that frequently change together. They are proposed as an alternative modular view, which can be used to assess the traditional decomposition of systems in packages. To investigate developer's perception of co-change clusters, we report a study with experts on six systems, implemented in two languages. We mine 102 co-change clusters from the version history of such systems, which are classified in three patterns regarding their projection to the package structure: Encapsulated, Crosscutting, and Octopus. We then collect the perception of expert developers on such clusters, aiming to ask two central questions: (a) what concerns and changes are captured by the extracted clusters? (b) do the extracted clusters reveal design anomalies? We conclude that Encapsulated Clusters are often viewed as healthy designs and that Crosscutting Clusters tend to be associated to design anomalies. Octopus Clusters are normally associated to expected class distributions, which are not easy to implement in an encapsulated way, according to the interviewed developers. [40]

## 7.4. Dynamic Languages: Debugging

**Practical domain-specific debuggers.** Understanding the run-time behavior of software systems can be a challenging activity. Debuggers are an essential category of tools used for this purpose as they give developers direct access to the running systems. Nevertheless, traditional debuggers rely on generic mechanisms to introspect and interact with the running systems, while developers reason about and formulate domain-specific questions using concepts and abstractions from their application domains. This mismatch creates an abstraction gap between the debugging needs and the debugging support leading to an inefficient and error-prone debugging effort, as developers need to recover concrete domain concepts using generic mechanisms. To reduce this gap, and increase the efficiency of the debugging process, we propose a framework for

developing domain-specific debuggers, called the Moldable Debugger, that enables debugging at the level of the application domain. The Moldable Debugger is adapted to a domain by creating and combining domain-specific debugging operations with domain-specific debugging views, and adapts itself to a domain by selecting, at run time, appropriate debugging operations and views. To ensure the proposed model has practical applicability (i.e., can be used in practice to build real debuggers), we discuss, from both a performance and usability point of view, three implementation strategies. We further motivate the need for domain-specific debugging, identify a set of key requirements and show how our approach improves debugging by adapting the debugger to several domains. [14]

**Mercury: Properties and Design of a Remote Debugging Solution using Reflection.** Remote debugging facilities are a technical necessity for devices that lack appropriate input/output interfaces (display, keyboard, mouse) for programming (e.g., smartphones, mobile robots) or are simply unreachable for local development (e.g., cloud-servers). Yet remote debugging solutions can prove awkward to use due to re-deployments. Empirical studies show us that on average 10.5 minutes per coding hour (over five 40-hour work weeks per year) are spent for redeploying applications (including re-deployments during debugging). Moreover current solutions lack facilities that would otherwise be available in a local setting because it is difficult to reproduce them remotely. Our work identifies three desirable properties that a remote debugging solution should exhibit, namely: run-time evolution, semantic instrumentation and adaptable distribution. Given these properties we propose and validate Mercury, a remote debugging model based on reflection. Mercury supports run-time evolution through a causally connected remote meta-level, semantic instrumentation through the reification of the underlying execution environment and adaptable distribution through a modular architecture of the debugging middleware. [19]

## 7.5. Reconciling Dynamic Languages and Isolation

**Handles.** Controlling object graphs and giving specific semantics to references (such as read-only, ownership, scoped sharing) have been the focus of a large body of research in the context of static type systems. Controlling references to single objects and to graphs of objects is essential to build more secure systems, but is notoriously hard to achieve in the absence of static type systems. In this article we embrace this challenge by proposing a solution to the following question: What is an underlying mechanism that can support the definition of properties (such as revocable, read-only, lent) at the reference level in the absence of a static type system? We present handles: first-class references that propagate behavioral change dynamically to the object subgraph during program execution. In this article we describe handles and show how handles support the implementation of read-only references and revocable references. Handles have been fully implemented by modifying an existing virtual machine and we report their costs. [13]

**Delegation Proxies.** Scoping behavioral variations to dynamic extents is useful to support non-functional concerns that otherwise result in cross-cutting code. Unfortunately, such forms of scoping are difficult to obtain with traditional reflection or aspects. We propose delegation proxies, a dynamic proxy model that supports behavioral intercession through the interception of various interpretation operations. Delegation proxies permit different behavioral variations to be easily composed together. We show how delegation proxies enable behavioral variations that can propagate to dynamic extents. We demonstrate our approach with examples of behavioral variations scoped to dynamic extents that help simplify code related to safety, reliability, and monitoring. [21]

**Access Control to Reflection with Object Ownership.** Reflection is a powerful programming language feature that enables language extensions, generic code, dynamic analyses , development tools, etc. However, uncontrolled reflection breaks object encapsulation and considerably increases the attack surface of programs e.g., malicious libraries can use reflection to attack their client applications. To bring reflection and object encapsulation back together, we use dynamic object ownership to design an access control policy to reflective operations. This policy grants objects full reflective power over the objects they own but limited reflective power over other objects. Code is still able to use advanced reflective operations but reflection cannot be used as an attack vector anymore. [41]

## 7.6. Tailoring Applications and boostrapping

**Virtualization Support for Dynamic Core Library Update**. Dynamically updating language runtime and core libraries such as collections and threading is challenging since the update mechanism uses such libraries at the same time that it modifies them. To tackle this challenge, we present Dynamic Core Library Update (DCU) as an extension of Dynamic Software Update (DSU) and our approach based on a virtualization architecture. Our solution supports the update of core libraries as any other normal library, avoiding the circular dependencies between the updater and the core libraries. Our benchmarks show that there is no evident performance overhead in comparison with a default execution. Finally, we show that our approach can be applied to real life scenario by introducing a critical update inside a web application with 20 simulated concurrent users. [34]

**Bootstrapping Infrastructure**. Bootstrapping is well known in the context of compilers, where a bootstrapped compiler can compile its own source code. Bootstrapping is a beneficial engineering practice because it raises the level of abstraction of a program making it easier to understand, optimize, evolve, etc. Bootstrapping a reflective object-oriented language is however more challenging , as we need also to initialize the runtime of the language with its initial objects and classes besides writing its compiler. We present a novel bootstrapping infrastructure for Pharo-like languages that allows us to easily extend and modify such languages. Our bootstrapping process relies on a first-class runtime. A first-class runtime is a meta-object that represents a program's runtime and provides a MOP to easily load code into it and manipulate its objects. It decou-ples the virtual machine (VM) and language concerns by introducing a clear VM-language interface. Using this process, we show how we succeeded to bootstrap a Smalltalk-based language named Candle and then extend it with traits in less than 250 lines of high-level Smalltalk code. We also show how we can bootstrap with minimal effort two other languages (Pharo and MetaTalk) with similar execution semantics but different object models. [35]

## 7.7. Dynamic Languages: Virtual Machines

**Towards Fully Reflective Environments.** Modern development environments promote live programming (LP) mechanisms because it enhances the development experience by providing instantaneous feedback and interaction with live objects. LP is typically supported with advanced reflective techniques within dynamic languages. These languages run on top of Virtual Machines (VMs) that are built in a static manner so that most of their components are bound at compile time. As a consequence, VM developers are forced to work using the traditional edit-compile-run cycle, even when they are designing LP-supporting environments. We explore the idea of bringing LP techniques to VM development to improve the observabil-ity, evolution and adaptability of VMs at run-time. We define the notion of fully reflective execution environments , systems that provide reflection not only at the application level but also at the level of the execution environment (EE). We characterize such systems, propose a design, and present Mate v1, a prototypical implementation. Based on our prototype, we analyze the feasibility and applicability of incorporating reflective capabilities into different parts of EEs. Furthermore, the evaluation demonstrates the opportunities such reflective capabilities provide for unanticipated dynamic adaptation scenarios, benefiting thus, a wider range of users. [23]

**Tracing vs. Partial Evaluation.** Tracing and partial evaluation have been proposed as meta-compilation techniques for interpreters to make just-in-time compilation language-independent. They promise that programs executing on simple interpreters can reach performance of the same order of magnitude as if they would be executed on state-of-the-art virtual machines with highly optimizing just-in-time compilers built for a specific language. Tracing and partial evaluation approach this meta-compilation from two ends of a spectrum, resulting in different sets of tradeoffs. This study investigates both approaches in the context of self-optimizing interpreters, a technique for building fast abstract-syntax-tree interpreters. Based on RPython for tracing and Truffle for partial evaluation, we assess the two approaches by comparing the impact of various optimizations on the performance of an interpreter for SOM, an object-oriented dynamically-typed language. The goal is to determine whether either approach yields clear performance or engineering benefits. We find that tracing and partial evaluation both reach roughly the same level of performance. SOM based on meta-tracing is on average 3x slower than Java, while SOM based on partial evaluation is on average 2.3x slower than Java. With

respect to the engineering, tracing has however significant benefits, because it requires language implementers to apply fewer optimizations to reach the same level of performance. [29]

**Zero-Overhead Metaprogramming.** Runtime metaprogramming enables many useful applications and is often a convenient solution to solve problems in a generic way, which makes it widely used in frameworks, middleware, and domain-specific languages. However, powerful metaobject protocols are rarely supported and even common concepts such as reflective method invocation or dynamic proxies are not optimized. Solutions proposed in literature either restrict the metaprogramming capabilities or require application or library developers to apply performance improving techniques. For overhead-free runtime metaprogramming, we demonstrate that dispatch chains, a generalized form of polymorphic inline caches common to self-optimizing interpreters, are a simple optimization at the language-implementation level. Our evaluation with self-optimizing interpreters shows that unrestricted metaobject protocols can be realized for the first time without runtime overhead, and that this optimization is applicable for just-in-time compilation of interpreters based on meta-tracing as well as partial evaluation. In this context, we also demonstrate that optimizing common reflective operations can lead to significant performance improvements for existing applications [30].

**A Partial Read Barrier for Efficient Support of Live Object-oriented Programming.** Live programming, originally introduced by Smalltalk and Lisp, and now gaining popularity in contemporary systems such as Swift, requires on-the-fly support for object schema migration, such that the layout of objects may be changed while the program is at one and the same time being run and developed. In Smalltalk schema migration is supported by two primitives, one that answers a collection of all instances of a class, and one that exchanges the identities of pairs of objects, called the become primitive. Existing instances are collected, copies using the new schema created, state copied from old to new, and the two exchanged with become, effecting the schema migration. Historically the implementation of become has either required an extra level of indirection between an object's address and its body, slowing down slot access, or has required a sweep of all objects, a very slow operation on large heaps. Spur, a new object representation and memory manager for Smalltalk-like languages, has neither of these deficiencies. It uses direct pointers but still provides a fast become operation in large heaps, thanks to forwarding objects that when read conceptually answer another object and a partial read barrier that avoids the cost of explicitly checking for forwarding objects on the vast majority of object accesses [31].

# 8. Bilateral Contracts and Grants with Industry

## 8.1. SafePython FUI

**Participants:** Damien Cassou [Correspondant], Jean-Baptiste Arnaud, Stéphane Ducasse.

**Contracting parties:** CEA, Evitech, Inria, Logilab, Opida, Thales, Wallix.

Beyond embedded computing, there is not so much research and development on the verification of software safety. Recently, some tools have been created for languages such as JAVA, SQL, VB or PHP. Nevertheless, nothing exists for Python even though this language is growing fast. SafePython's goal is to provide code analysis tools applicable to Python programs. This project will define a subset of Python that the developers will have to use to have their programs analyzed.

## 8.2. Sponsoring LAM

**Participants:** Stéphane Ducasse [Correspondant], Marcus Denker.

**Contracting parties:** Inria, LAM Research, Inc.

LAM Research Inc.(http://lamrc.com) is a leading supplier of wafer fabrication equipment and services to the global semiconductor industry. LAM has started to sponsor RMOD in 2014. RMOD used the sponsored funds to pay student internships in 2015.

## 8.3. Worldline CIFRE

**Participants:** Anne Etien [Correspondant], Nicolas Anquetil, Stéphane Ducasse, Vincent Blondeau.

In the context of a CIFRE PhD we are working on large industrial project characterization. The PhD started in October 2014.

## 8.4. Pharo Consortium

The Pharo Consortium was founded in 2012 and is growing constantly. As of end 2015, it has 19 company members, 13 academic partners and 3 sponsoring companies. Inria supports the consortium with one full time engineer starting in 2011. More at http://consortium.pharo.org.

# 9. Partnerships and Cooperations

## 9.1. Regional Initiatives

We have signed a convention with the CAR team led by Noury Bouraqadi of Ecole des Mines de Douai. In this context we co-supervized three PhD students (Mariano Martinez-Peck, Nick Papoylias and Guillermo Polito). The team is also an important contributor and supporting organization of the Pharo project.

## 9.2. European Initiatives

### *9.2.1. FP7 & H2020 Projects*

**MEALS FP7 Marie Curie Research Staff Exchange Scheme**

MEALS (Mobility between Europe and Argentina applying Logics to Systems) is a mobility project financed by the 7th Framework programme under the Marie Curie International Research Staff Exchange Scheme. It involves seven academic institutions from Europe and four from Argentina, and a total of about 80 researchers to be exchanged. The project started on the 1st of October, 2011, and it has a duration of 4 years. Nr: FP7-PEOPEL-2011-IRSES

http://www.meals-project.eu

### *9.2.2. Collaborations in European Programs, except FP7 & H2020*

*9.2.2.1. ERCIM Software Evolution*

We are involved in the ERCIM Software Evolution working group since its inception. We participated at his creation when we were at the University of Bern.

## 9.3. International Initiatives

### *9.3.1. Inria International Labs*

**Inria Chile**
Associate Team involved in the International Lab:

#### *9.3.1.1. PLOMO2*

Title: Infrastructure for a new generation of development tools

International Partner (Institution - Laboratory - Researcher):

Universidad de Chile (Chile) - Computer Science Department, PLEIAD laboratory (DCC) - Alexander Bergel

Start year: 2014

See also: http://pleiad.cl/research/plomo2

Performing effective software development and maintenance are best achieved with effective tool support. Provided by a variety of tools, each one presenting a specific kinds of information supporting the task at hand. With Plomo2, we want to invent a new generation tools to navigate and profile programs by combining dynamic information with visualization to improve the development environment.

### *9.3.2. Inria International Partners*

#### *9.3.2.1. Declared Inria International Partners*

**Participants:** Marcus Denker [correspondant], Stéphane Ducasse [RMoD], Nicolas Anquetil [RMoD], Diego Garbervetsky [UBA,LAFHIS], Gabriela Arevalo [Universidad Nacional de Quilmes)], Nicolas Passerini [Uqbar].

**Uqbar - Argentina**

Uqbar is a foundation of researchers teaching in several universities of the Buenos Aires area. Universidad Tecnologica Nacional (FRBA) Universidad Nacional de Quilmes, Universidad Nacional de San Martin,Universidad Nacional del Oeste. LAFHIS is a research laboratory from the University of Buenos Aires. More information at (http://www.uqbar-project.org).

#### *9.3.2.2. Informal International Partners*

**Pharo in Research:** We are building an ecosystem around Pharo with international research groups, universities and companies. Several research groups (such as Software Composition Group – Bern, and Pleaid – Santiago) are using Pharo. Many universities are teaching OOP using Pharo and its books. Several companies worldwide are deploying business solutions using Pharo.

**University of Mons, Belgium** Julien Delplanque is a student in the master M1 program from Mons University. He is working on SQL parsers and code critics.

### *9.3.3. Participation In other International Programs*

#### *9.3.3.1. STIC AmSud*

**Participants:** Damien Cassou [correspondant], Gustavo Santos [RMoD], Martin Dias [RMoD], David Röthlisberger [UDP - Universidad Diego Portales, Santiago, Chile], Marcelo Almeida Maia [UFU - Federal University of Uberlândia, Brasil], Romain Robbes [Departamento de Ciencias de la Computación (DCC), Universidad de Chile, Santiago, Chile], Martin Monperrus [Spirals].

Project Partners: Inria RMOD, Inria Spirals, DCC Universidad de Chile, Universidad Diego Portale Chile, Federal University of Uberlândia, Brasil.

This project aims at facilitating the usage of frameworks and application programming interfaces (APIs) by mining software repositories. Our intuition is that mining reveals how existing projects instantiate these frameworks. By locating concrete framework instantiations in existing projects, we can recommend to developers the concrete procedures for how to use a particular framework for a particular task in a new system. Our project also tackles the challenge of adapting existing systems to new versions of a framework or API by seeking repositories for how other systems adapted to such changes. We plan to integrate recommendations of how to instantiate a framework and adapt to changes directly in the development environment. Those points taken together, considerably distinguish our approach from existing research in the area of framework engineering.

Nicolas Anquetil visited one week the ASERG team of Pr. Marco Tulio Valente at Federal University of Minas Gerais (Brazil), and another week the team of Pr. Alexander Bergel at University of Chile.

*9.3.3.2. European Lab with Delft*

We have a Lille Nord Europe European Lab with A. Bachelli from Delft University. We are working on infrastructure and tools for code reviewing. We have exchange of staff and presented a paper at SANER 2015.

## 9.4. International Research Visitors

### *9.4.1. Visits of International Scientists*

In the context of the PLOMO2 associated Team with the University of Chile:

- Johan Fabry (January 2015 for Pharo Days Lille, PLEIAD funded)
- Alexandre Bergel (01/07/2015 until 27/07/2015)
- Johan Fabry (July 2015 for ESUG, PLEIAD funded)
- Pierre Chanson (July 2015 and September 2015)
- Miguel Campusano (20/09/2015-07/10/2015)
- Alexandre Bergel (Dec 2015)

In the context of MEALS:

- Guido Chari visited RMoD in March 2015.

Other visitors:

- Glenn Cavarle, Jun 2015. Pharo MOOC.
- Skip Lentz, Delft University of Technology, Delft, the Netherlands, September 2015 to January 2016. Internship/Research project.
- Matthieu Lacaton, Thales, April 2015.
- Alain Plantec, Univ. Bretagne Occidentale, until June 2015. Bloc UI Project.
- Klérisson Vinícius Ribeiro da Paixão, Federal University of Uberlândia, Uberlândia (MG), Brazil, from September, 2015 to July, 2016. Stic-Amsud MineAPI.
- Markiyan Rizun, Ivan Franko National University of Lviv, Ukraine, From 11 January 2015 to 5 February 2015 and 4 July 2015 7 August 2015. Rewrite Tool.
- Nicolás Passerini, Uqbar / Universidad Nacional de Quilmes, Buenos Aires, Argentina, 24/11/2015 to 27/11/2015.
- Pablo Tesone, Thesis Relay from August 2015.
- Abdelghani ALIDRA, University 20 Aout 55, Skikda. Algeria. One month in June-July and November 2015 to December 2016.
- Leonardo Silva, PhD student from Brazil (Federal University of Minas Gerais), did a 6 months internship within for his PhD. From January 2015 to June 2015. SticAmsud Project Dynarchi.

### *9.4.2. Visits to International Teams*

*9.4.2.1. Research stays abroad*

- Guillermo Polito: 25.05.2015 SOFT Languages Lab Vrije Universiteit Brussel, Brussels. Visit to present the work of the laboratory and look for collaborations.
- Guillermo Polito: 23.07.2015 Universidad de Quilmes, Buenos Aires. Visit to present the work of the laboratory.
- Nicolas Anquetil visited one week the ASERG team of Pr. Marco Tulio Valente at Federal University of Minas Gerais (Brazil), and another week the team of Pr. Alexander Bergel at University of Chile.

- Marcus Denker: 02.11-02.12.2015 PLEIAD DCC University of Chile, Santiago de Chile. Visit in the context of the Inria Associated Team PLOMO2.
- Marcus Denker: 02.01-20.01.2015 PLEIAD DCC University of Chile, Santiago de Chile. Visit in the context of the Inria Associated Team PLOMO2.
- Damien Cassou and Gustavo Santos. 02.02.2015. Visited for one week the University of Uberlandia (Brazil). Project MineAPI (SticAmSud).
- Olivier Auverlot and Anne Etien 16.12.15. University of Namur, Belgium. Visit to present the SQL project.

# 10. Dissemination

## 10.1. Promoting Scientific Activities

### *10.1.1. Scientific events organisation*

*10.1.1.1. General chair, scientific chair*

Anne Etien was General chair of IWST15, the International Workshop of Smalltalk Technologies (part of ESUG 2015).

*10.1.1.2. Member of the organizing committees*

- Nicolas Anquetil, Anne Etien and Damien Cassou organized the umbrella conference EvoLille (= EOSESE + RIMEL + BENEVOL) as well as BENEVOL. Funding was aquired from sponsors (8000 EUR). The event had 60 participants.
- Nicolas Anquetil organized RIMEL.
- Marcus Denker and Stéphane Ducasse are in the boad of ESUG and organized ESUG 2015, the yearly Smalltalk industrial conference.
- RMoD organized Pharo Conf 2015, the yearly Pharo meeting of both industrial and reserach users.

### *10.1.2. Scientific events selection*

*10.1.2.1. Chair of conference program committees*

- Nicolas Anquetil, Anne Etien and Damien Cassou where program chairs of BENEVOL, BElgian-NEtherlands software eVOLution seminar (Dec 03-04, 2015. Lille, France)
- Anne Etien was program chair at IWST 2015 (International Workshop of Smalltalk Technologies).

*10.1.2.2. Member of the conference program committees*

- Anne Etien
  - BENEVOL 2015 : BElgian-NEtherlands software eVOLution seminar (Dec 03-04, 2015. Lille, France)
  - Modelsward16 : 4th International Conference on Model-Driven Engineering and Software Development (19 - 21 Feb 2016, Rome, Italy)
  - ICSOFT-PT15 : 10th International Conference on Software Paradigm Trends
  - ICSOFT-PT16 : 11th International Conference on Software Paradigm Trends (24 - 26 Jul 2016, Lisbon, Partugal)
- Nicolas Anquetil
  - BENEVOL 2015 : BENEVOL 2015 : BElgian-NEtherlands software eVOLution seminar (Dec 03-04, 2015. Lille, France)
  - SANER 2015 : International Conference on Software Analysis, Evolution, and Reengineering (Mar 2–6, 2015; Montreal, Canada)

  - CIEL 2015 : 4 me Conference en Ingenierie du Logiciel (Jun 9–11, 2015; Bordeaux, France)
  - ICSOFT-EA : 10th International Joint Conference on Software Engineering and Applications (Jul 20–22, 2015; Colmar, France)
  - AICCSA 2015 : ACS/IEEE International Conference on Computer Systems and Applications (Nov 17–20, 2015; Marrakech, Morocco)
- Damien Cassou
  - BENEVOL 2015 : BElgian-NEtherlands software eVOLution seminar (Dec 03-04, 2015. Lille, France)
- Marcus Denker
  - SANER 2016 (23rd IEEE International Conference on Software Analysis, Evolution, and Reengineering)
  - VISSOFT'15 NIER/TD (IEEE Working Conference on Software Visualization, NIER and Tool Track)
- Martin Dias
  - DChanges 2015 (International Workshop on (Document) Changes: Modeling, Detection, Storage and Visualization 2015)
- Stéphane Ducasse
  - International Conference Onward! 2015.
  - International Conference 18th International Conference on Fundamental Approaches to Software Engineering (FASE) 2015, 2016
  - declined: ECOOP 2015, ECOOP 2016.

*10.1.2.3. Reviewer*

- Gustavo Santos reviewed for SANER 2016 (23rd IEEE International Conference on Software Analysis, Evolution, and Reengineering).
- Klérisson Paixão reviewed for SANER 2016 (23rd IEEE International Conference on Software Analysis, Evolution, and Reengineering).
- Klérisson Paixão reviewed for 19th International Conference on Fundamental Approaches to Software Engineering (FASE).

### *10.1.3. Journal*

*10.1.3.1. Reviewer - Reviewing activities*

Nicolas Anquetil reviewed a book on software evolution for Springer. He also reviewed manuscripts for the following intyernational journals:

- IEEE Transactions on Software Engineering (IEEE Computer Society, ISSN: 0098-5589)
- Journal of Software: Evolution and Process (Wiley, ISSN: 2047-7481)
- Journal of Systems and Software (Elsevier, ISSN: 0164-1212)
- IET Software (IET, ISSN 1751-8806)

Marcus Denker reviewed an article for Science of Computer Programming (Elsevier, ISSN: 0167-6423)

### *10.1.4. Scientific expertise*

- Damien Cassou reviewed a proposal for *Jeune Entreprise Innovante*.
- Marcus Denker reviewed a research proposal for *Fondo Nacional de Desarrollo Cientifico y Tecnologico (FONDECYT), Chile*

### *10.1.5. Research administration*

- Marcus Denker and Stéphane Ducasse are Moderators of arXiv cs.SE, from 2015. http://arxiv.org/list/cs.SE/recent
- Marcus Denker was vice president of the local Inria Lille *commission développement technologique*.

## 10.2. Teaching - Supervision - Juries

### *10.2.1. Teaching*

Master: C. Demarey, Intégration Continue, 4h (M2), GIS2A5, Polytech'Lille, France

Master: C. Demarey, Intégration Continue, 12h (M2), IAGL, Université de Lille 1, France

Licence : Damien Cassou, OpenDevs Communauté de Développeurs, 40, L3, Université Lille 1, France

Licence : Damien Cassou, Conception orientée objet, 52.5, L3, Université Lille 1, France

Master : Damien Cassou, Qualité logicielle, 30, M2, Université Lille 1, France

Master : Damien Cassou, CAL Tests et Maintenance, 12, M2, Université Lille 1, France

Master: Anne Etien, Test et Maintenance, 10, M2, Polytech-Lille, Frabce

Master : Anne Etien, Syste`me d'information objet, 22, M1, Polytech-Lille, France

Master : Anne Etien, Bases de donne´es, 44, M1, Polytech-Lille, France

Master : Anne Etien, Bases de donne´es Avance´es, 12, M1, Polytech-Lille, France

Licence : Anne Etien, Bases de donne´es, 22, L3, Polytech-Lille, France

Licence : Damien Pollet, Java lecture, L3 level, 130 students, 30 hours (for the students), Telecom Lille, France

Licence : Damien Pollet, Java lecture, L3 level, 60 remote students, 20h, Telecom Lille, France

Licence : Damien Pollet, Java project, L3 level, 10 remote students, 20h, Telecom Lille, France

Licence : Damien Pollet, Computer architecture, L3 level, 10h, Telecom Lille, France

Master : Damien Pollet, Technologies for information systems, M1 level 30h, Telecom Lille, France

Master : Damien Pollet, Network algorithms, M2, 30h, Telecom Lille, France

Master : Damien Pollet, Software engineering, M1, 6h, Telecom Lille, France

Nicolas Anquetil, Conception et programmation objet avancées, 64h, L2, Univ. Lille1, France

Nicolas Anquetil, Programation Système, 48h L2, Univ. Lille1, France

Nicolas Anquetil, Projet tuteuré, 32h L2, univ. Lille1, France

Nicolas Anquetil, Programmation mobile, 32h L2, univ. Lille1, France

### *10.2.2. Supervision*

Martin Dias, Supporting Merging, 27/11/2015, Damien Cassou, Stéphane Ducasse.

Guillermo Polito, Isolation and Reflection in Dynamic Object Oriented Languages, 13/04/2015, Noury Bouraqadi, Luc Fabresse, Stéphane Ducasse.

PhD in progress: Vincent Blondeau, CIFRE Worldline, started Oct 2014, Anne Etien, Nicolas Anquetil.

PhD in progress: Brice Govin, CIFRE Thales, started Jan 2015, Anne Etien, Nicolas Anquetil, Stéphane Ducasse.

PhD in progress: Gustavo Santos, started April 2014, Anne Etien, Nicolas Anquetil

PhD in progress: Camille Teruel, Security for dynamic languages, 01/12/2012, Stéphane Ducasse, Damien Cassou

PhD in progress: Marco Naddeo, Evolvable and reusable software: from language design to developer tools, started 01/01/2014, Viviana Bono (University of Turin), Damien Cassou, Stéphane Ducasse

PhD in progress: Clément Béra: Evolvable Runtimes for a Changing World, started Oct 2014, Stéphane Ducasse, Marcus Denker.

PhD in progresss: Klérisson Paixao, Automating reactions for API changes based on dynamic analysis, started 16/08/2014, Marcelo Maia, Damien Cassou, Stéphane Ducasse.

Internships: Cyril Ferlicot, Yann Lesage, Skip Lentz, Thomas Durieux, Cédric Lenquette, Xavier Koma, Mathieu Schepens, Thomas Heniart.

### *10.2.3. Juries*

- Anne Etien: Eli Richa, Qualification des générateurs de code source dans le domaine de l'avionique: le test automatisé des chaines de transformation de modèles, 15/12/15, Telecom ParisTech (reviewer)
- Damine Cassou: Martin Dias, Supporting Integration Activities with First-Class Code Changes, 27/11/2015, Lille.
- Stéphane Ducasse:
  - *Habilitation*
    - *Reflective Languages for Mobile Robotics Applications Development*, Dr. Luc Fabresse, Université de Lille 1, Lille, 08/12/15.
    - *Supporting Software Evolution in Organizations*, Dr. Nicolas Anquetil, Université de Lille 1, Lille, 09/05/14.
  - *Thèses*
    - *Organisation des développeurs open-source et fiabilité logicielle, Mathieu Foucault*, Université de Bordeaux, France. 30/11/15. (referee)

## 10.3. Popularization

- Marcus Denker gave one presentation about Pharo at the open source conference FOSDEM 2015.
- RMOD co-organized and participated at ESUG 2015.
- Multiple public Pharo Sprints in Lille.
- A MOOC for Pharo is in preparation (Stéphane Ducasse, Damien Cassou)

# 11. Bibliography

## Major publications by the team in recent years

[1] N. ANQUETIL, K. M. DE OLIVEIRA, K. D. DE SOUSA, M. G. BATISTA DIAS. *Software maintenance seen as a knowledge management issue*, in "Information Software Technology", 2007, vol. 49, n$^{o}$ 5, pp. 515–529 [*DOI :* 10.1016/J.INFSOF.2006.07.007], http://rmod.lille.inria.fr/archives/papers/Anqu07a-IST-MaintenanceKnowledge.pdf

[2] M. DENKER, S. DUCASSE, É. TANTER. *Runtime Bytecode Transformation for Smalltalk*, in "Journal of Computer Languages, Systems and Structures", July 2006, vol. 32, n$^{o}$ 2-3, pp. 125–139 [*DOI : 10.1016/J.CL.2005.10.002*], http://rmod.lille.inria.fr/archives/papers/Denk06a-COMLAN-RuntimeByteCode.pdf

[3] S. DUCASSE, O. NIERSTRASZ, N. SCHÄRLI, R. WUYTS, A. P. BLACK. *Traits: A Mechanism for fine-grained Reuse*, in "ACM Transactions on Programming Languages and Systems (TOPLAS)", March 2006, vol. 28, n$^{o}$ 2, pp. 331–388 [*DOI : 10.1145/1119479.1119483*], http://scg.unibe.ch/archive/papers/Duca06bTOPLASTraits.pdf

[4] S. DUCASSE, D. POLLET. *Software Architecture Reconstruction: A Process-Oriented Taxonomy*, in "IEEE Transactions on Software Engineering", July 2009, vol. 35, n$^{o}$ 4, pp. 573-591 [*DOI : 10.1109/TSE.2009.19*], http://rmod.lille.inria.fr/archives/papers/Duca09c-TSE-SOAArchitectureExtraction.pdf

[5] S. DUCASSE, D. POLLET, M. SUEN, H. ABDEEN, I. ALLOUI. *Package Surface Blueprints: Visually Supporting the Understanding of Package Relationships*, in "ICSM'07: Proceedings of the IEEE International Conference on Software Maintenance", 2007, pp. 94–103, http://scg.unibe.ch/archive/papers/Duca07cPackageBlueprintICSM2007.pdf

[6] A. KUHN, S. DUCASSE, T. GÎRBA. *Semantic Clustering: Identifying Topics in Source Code*, in "Information and Software Technology", March 2007, vol. 49, n$^{o}$ 3, pp. 230–243 [*DOI : 10.1016/J.INFSOF.2006.10.017*], http://scg.unibe.ch/archive/drafts/Kuhn06bSemanticClustering.pdf

[7] J. LAVAL, S. DENIER, S. DUCASSE, A. BERGEL. *Identifying cycle causes with Enriched Dependency Structural Matrix*, in "WCRE '09: Proceedings of the 2009 16th Working Conference on Reverse Engineering", Lille, France, 2009, http://rmod.lille.inria.fr/archives/papers/Lava09c-WCRE2009-eDSM.pdf

[8] O. NIERSTRASZ, S. DUCASSE, T. GÎRBA. *The Story of Moose: an Agile Reengineering Environment*, in "Proceedings of the European Software Engineering Conference", New York NY, M. WERMELINGER, H. GALL (editors), ESEC/FSE'05, ACM Press, 2005, pp. 1–10, Invited paper [*DOI : 10.1145/1095430.1081707*], http://scg.unibe.ch/archive/papers/Nier05cStoryOfMoose.pdf

[9] J. SINGER, T. LETHBRIDGE, N. VINSON, N. ANQUETIL. *An examination of software engineering work practices*, in "Proceedings of the 1997 conference of the Centre for Advanced Studies on Collaborative research", CASCON '97, IBM Press, 1997, http://rmod.lille.inria.fr/archives/papers/Sing97a-SoftwareEngineeringWorkPractices.pdf

[10] S. C. B. DE SOUZA, N. ANQUETIL, K. M. DE OLIVEIRA. *A study of the documentation essential to software maintenance*, in "Proceedings of the 23rd annual international conference on Design of communication: documenting & designing for pervasive information", New York, NY, USA, SIGDOC '05, ACM, 2005, pp. 68–75, http://dx.doi.org/10.1145/1085313.1085331

## Publications of the year

### Doctoral Dissertations and Habilitation Theses

[11] M. DIAS. *Supporting Software Integration Activities with First-Class Code Changes*, Laboratoire d'Informatique Fondamentale de Lille, November 2015, https://hal.inria.fr/tel-01247696

[12] G. POLITO. *Virtualization Support for Application Runtime Specialization and Extension*, Universite des Sciences et Technologies de Lille, April 2015, https://hal.inria.fr/tel-01251173

### Articles in International Peer-Reviewed Journals

[13] J.-B. ARNAUD, S. DUCASSE, M. DENKER, C. TERUEL. *Handles: Behavior-Propagating First Class References For Dynamically-Typed Languages*, in "Journal of Science of Computer Programming", February 2015, vol. 98, n° 3, pp. 318–338 [*DOI :* 10.1016/J.SCICO.2014.07.011], https://hal.inria.fr/hal-01060537

[14] A. CHIŞ, M. DENKER, T. GÎRBA, O. NIERSTRASZ. *Practical domain-specific debuggers using the Moldable Debugger framework*, in "Computer Languages, Systems and Structures", December 2015, vol. 44, n° Part A, pp. 89-113 [*DOI :* 10.1016/J.CL.2015.08.005], https://hal.inria.fr/hal-01247941

[15] M. DE WAEL, S. MARR, B. DE FRAINE, T. VAN CUTSEM, W. DE MEUTER. *Partitioned Global Address Space Languages*, in "ACM Computing Surveys", January 2016, 29 p. , https://hal.inria.fr/hal-01109405

[16] A. HORA, N. ANQUETIL, A. ETIEN, S. DUCASSE, M. T. VALENTE. *Automatic Detection of System-Specific Conventions Unknown to Developers*, in "Journal of Systems and Software", August 2015, vol. 109, pp. 192–204 [*DOI :* 10.1016/J.JSS.2015.08.007], https://hal.inria.fr/hal-01185837

[17] C. MAFFORT, M. T. VALENTE, R. TERRA, M. BIGONHA, N. ANQUETIL, A. HORA. *Mining Architectural Violations from Version History*, in "Empirical Software Engineering", 2015, 41 p. , https://hal.inria.fr/hal-01075642

[18] M. MARTINEZ PECK, N. BOURAQADI, L. FABRESSE, M. DENKER, C. TERUEL. *Ghost: A uniform and general-purpose proxy implementation*, in "Science of Computer Programming", February 2015, vol. 98, n° 3, pp. 339-359 [*DOI :* 10.1016/J.SCICO.2014.05.015], https://hal.inria.fr/hal-01081236

[19] N. PAPOULIAS, N. BOURAQADI, L. FABRESSE, S. DUCASSE, M. DENKER. *Mercury: Properties and Design of a Remote Debugging Solution using Reflection*, in "Journal of Object Technology", September 2015, vol. 14, n° 2, 36 p. [*DOI :* 10.5381/JOT.2015.14.2.A1], https://hal.inria.fr/hal-01185730

[20] P. PATEL, D. CASSOU. *Enabling High-Level Application Development for the Internet of Things*, in "Journal of Systems and Software", 2015, pp. 1 - 26, https://hal.inria.fr/hal-01107498

[21] C. TERUEL, E. WERNLI, S. DUCASSE, O. NIERSTRASZ. *Propagation of Behavioral Variations with Delegation Proxies*, in "Transactions on Aspect-Oriented Software Development (TAOSD)", 2015, vol. 8989, pp. 63-95 [*DOI :* 10.1007/978-3-662-46734-3_2], https://hal.archives-ouvertes.fr/hal-01135706

### International Conferences with Proceedings

[22] V. BLONDEAU, N. ANQUETIL, S. DUCASSE, S. CRESSON, P. CROISY. *Software metrics to predict the health of a project ?*, in "IWST '15 International Workshop On Smalltalk Technologies", Brescia, Italy, ACM, July 2015, 8 p. [*DOI :* 10.1145/2811237.2811294], https://hal.inria.fr/hal-01185079

[23] G. CHARI, D. GARBERVETSKY, S. MARR, S. DUCASSE. *Towards Fully Reflective Environments*, in "Onward!", Pittsburg, France, October 2015, https://hal.inria.fr/hal-01185843

[24] M. DE WAEL, S. MARR, J. DE KOSTER, J. B. SARTOR, W. DE MEUTER. *Just-in-Time Data Structures*, in "Proceedings of the 2015 ACM International Symposium on New Ideas, New Paradigms, and Reflections on Programming & Software", Pittsburgh, PA, United States, October 2015 [*DOI :* 10.1145/2814228.2814231], https://hal.inria.fr/hal-01205343

[25] M. DIAS, A. BACCHELLI, G. GOUSIOS, D. CASSOU, S. DUCASSE. *Untangling Fine-Grained Code Changes*, in "SANER: International Conference on Software Analysis, Evolution, and Reengineering", Montréal, Canada, March 2015, https://hal.inria.fr/hal-01116225

[26] M. DIAS, G. POLITO, D. CASSOU, S. DUCASSE. *DeltaImpactFinder: Assessing Semantic Merge Conflicts with Dependency Analysis*, in "International Workshop on Smalltalk Technologies 2015", Brescia, Italy, ESUG, July 2015 [*DOI :* 10.1145/2811237.2811299], https://hal.inria.fr/hal-01199035

[27] A. HORA, R. ROBBES, N. ANQUETIL, A. ETIEN, S. DUCASSE, M. T. VALENTE. *How Do Developers React to API Evolution? The Pharo Ecosystem Case*, in "31st IEEE International Conference on Software Maintenance", Bremen, Germany, September 2015, 10 p. , https://hal.inria.fr/hal-01185736

[28] L. HUMBERTO SILVA, M. RAMOS, M. T. VALENTE, A. BERGEL, N. ANQUETIL. *Does JavaScript software embrace classes?*, in "SANER 2015 : International Conference on Software Analysis, Evolution, and Reengineering", Montreal, Canada, March 2015, pp. 73 - 82 [*DOI :* 10.1109/SANER.2015.7081817], https://hal.inria.fr/hal-01185854

[29] S. MARR, S. DUCASSE. *Tracing vs. Partial Evaluation*, in "Proceedings of ACM International Conference on Object Oriented Programming Systems Languages & Applications (OOPSLA '15)", Pittsburgh, PA, United States, October 2015 [*DOI :* 10.1145/2814270.2814275], https://hal.inria.fr/hal-01205345

[30] S. MARR, C. SEATON, S. DUCASSE. *Zero-Overhead Metaprogramming*, in "Proceedings of the 36th ACM SIGPLAN Conference on Programming Language Design and Implementation", Portland, OR, USA, France, June 2015 [*DOI :* 10.1145/2737924.2737963], https://hal.inria.fr/hal-01141135

[31] E. MIRANDA, C. BÉRA. *A Partial Read Barrier for Efficient Support of Live Object-oriented Programming*, in "International Symposium on Memory Management", Portland, United States, ISMM '15 Proceedings of the 2015 International Symposium on Memory Management, June 2015, pp. 93-104 [*DOI :* 10.1145/2754169.2754186], https://hal.inria.fr/hal-01152610

[32] H. OUMAROU, N. ANQUETIL, A. ETIEN, S. DUCASSE, D. KOLYANG. *Identifying the exact fixing actions of static rule violation*, in "SANER'15 : 22nd IEEE International Conference on Software Analysis, Evolution, and Reengineering", Montreal, Canada, February 2015, n$^{o}$ 371 - 379 [*DOI :* 10.1109/SANER.2015.7081847], https://hal.inria.fr/hal-01185795

[33] N. PAPOULIAS, M. DENKER, S. DUCASSE, L. FABRESSE. *Reifying the Reflectogram*, in "30th ACM/SIGAPP Symposium On Applied Computing", Salamanca, Spain, April 2015 [*DOI :* 10.1145/2695664.2695883], https://hal.inria.fr/hal-01098596

[34] G. POLITO, S. DUCASSE, N. BOURAQADI, L. FABRESSE, M. MATTONE. *Virtualization Support for Dynamic Core Library Update*, in "Onward!", Pittsburg, United States, October 2015 [*DOI :* 10.1145/2814228.2814236], https://hal.inria.fr/hal-01185819

[35] G. POLITO, S. DUCASSE, L. FABRESSE, N. BOURAQADI. *A Bootstrapping Infrastructure to Build and Extend Pharo-Like Languages*, in "Onward!", Pittsburg, United States, June 2015 [*DOI :* 10.1145/2814228.2814236], https://hal.inria.fr/hal-01185812

[36] D. POLLET, S. DUCASSE. *A First Analysis of String APIs: the Case of Pharo*, in "IWST '15 International Workshop On Smalltalk Technologies", Brescia, Italy, ACM, June 2015 [*DOI :* 10.1145/2811237.2811298], https://hal.archives-ouvertes.fr/hal-01244486

[37] G. SANTOS, N. ANQUETIL, A. ETIEN, S. DUCASSE, M. TULIO VALENTE. *OrionPlanning: Improving Modularization and Checking Consistency on Software Architecture*, in "3rd IEEE Working Conference on Software Visualization (VISSOFT)", Bremen, Germany, September 2015, 5 p. , https://hal.inria.fr/hal-01185635

[38] G. SANTOS, N. ANQUETIL, A. ETIEN, S. DUCASSE, M. TULIO VALENTE. *System Specific, Source Code Transformations*, in "31st IEEE International Conference on Software Maintenance and Evolution (ICSME)", Bremen, Germany, September 2015, 10 p. , https://hal.inria.fr/hal-01185637

[39] G. SANTOS, A. ETIEN, N. ANQUETIL, S. DUCASSE, M. TULIO VALENTE. *Recording and Replaying System Specific, Source Code Transformations*, in "15th IEEE International Working Conference on Source Code Analysis and Manipulation (SCAM)", Bremen, Germany, September 2015, 10 p. , https://hal.inria.fr/hal-01185639

[40] L. L. SILVA, M. T. VALENTE, M. MAIA, N. ANQUETIL. *Developers' Perception of Co-Change Patterns: An Empirical Study*, in "Proceedings of the 31st IEEE International Conference on Software Maintenance", Bremen, Germany, September 2015, https://hal.inria.fr/hal-01185865

[41] C. TERUEL, S. DUCASSE, D. CASSOU, M. DENKER. *Access Control to Reflection with Object Ownership*, in "Dynamic Languages Symposium", USA, France, Proceedings of the 11th Symposium on Dynamic Languages, October 2015, pp. 168-176 [*DOI :* 10.1145/2816707.2816721], https://hal.inria.fr/hal-01217041

### Conferences without Proceedings

[42] V. BLONDEAU, S. CRESSON, P. CROISY, A. ETIEN, N. ANQUETIL, S. DUCASSE. *Predicting the health of a project? An assessment in a major IT company*, in "SATToSE'15", Mons, Belgium, July 2015, https://hal.inria.fr/hal-01205468

[43] B. GOVIN, N. ANQUETIL, A. ETIEN, A. MONEGIER DU SORBIER, S. DUCASSE. *Reverse Engineering Tool Requirements for Real Time Embedded Systems*, in "SATToSE'15", Mons, Belgium, July 2015, https://hal.inria.fr/hal-01187532

### Scientific Books (or Scientific Book chapters)

[44] D. CASSOU, S. DUCASSE, L. FABRESSE, J. FABRY, S. VAN CAEKENBERGHE. *Enterprise Pharo a Web Perspective*, Square Bracket Associates, November 2015, 279 p. , https://hal.inria.fr/hal-01223026

### Research Reports

[45] M. DENKER, N. ANQUETIL, D. CASSOU, S. DUCASSE, A. ETIEN, D. POLLET. *Project-Team RMoD 2014 Activity Report*, Inria Lille ; RMOD, January 2015, https://hal.inria.fr/hal-01247323

### Other Publications

[46] N. PAPOULIAS. *Seamless: A Reflective Middleware for Pharo (DRAFT)*, April 2015, working paper or preprint, https://hal.inria.fr/hal-01145792

## References in notes

[47] N. ANQUETIL. *A Comparison of Graphs of Concept for Reverse Engineering*, in "Proceedings of the 8th International Workshop on Program Comprehension", Washington, DC, USA, IWPC '00, IEEE Computer Society, 2000, pp. 231–, http://rmod.lille.inria.fr/archives/papers/Anqu00b-ICSM-GraphsConcepts.pdf

[48] A. BERGEL, S. DUCASSE, O. NIERSTRASZ. *Classbox/J: Controlling the Scope of Change in Java*, in "Proceedings of 20th International Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA'05)", New York, NY, USA, ACM Press, 2005, pp. 177–189 [*DOI :* 10.1145/1094811.1094826], http://scg.unibe.ch/archive/papers/Berg05bclassboxjOOPSLA.pdf

[49] A. BERGEL, S. DUCASSE, O. NIERSTRASZ, R. WUYTS. *Stateful Traits*, in "Advances in Smalltalk — Proceedings of 14th International Smalltalk Conference (ISC 2006)", LNCS, Springer, August 2007, vol. 4406, pp. 66–90, http://dx.doi.org/10.1007/978-3-540-71836-9_3

[50] A. BERGEL, S. DUCASSE, O. NIERSTRASZ, R. WUYTS. *Stateful Traits and their Formalization*, in "Journal of Computer Languages, Systems and Structures", 2008, vol. 34, n° 2-3, pp. 83–108, http://dx.doi.org/10.1016/j.cl.2007.05.003

[51] A. P. BLACK, N. SCHÄRLI, S. DUCASSE. *Applying Traits to the Smalltalk Collection Hierarchy*, in "Proceedings of 17th International Conference on Object-Oriented Programming Systems, Languages and Applications (OOPSLA'03)", October 2003, vol. 38, pp. 47–64 [*DOI :* 10.1145/949305.949311], http://scg.unibe.ch/archive/papers/Blac03aTraitsHierarchy.pdf

[52] G. BRACHA, D. UNGAR. *Mirrors: design principles for meta-level facilities of object-oriented programming languages*, in "Proceedings of the International Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA'04), ACM SIGPLAN Notices", New York, NY, USA, ACM Press, 2004, pp. 331–344, http://bracha.org/mirrors.pdf

[53] D. CAROMEL, J. VAYSSIÈRE. *Reflections on MOPs, Components, and Java Security*, in "ECOOP '01: Proceedings of the 15th European Conference on Object-Oriented Programming", Springer-Verlag, 2001, pp. 256–274

[54] D. CAROMEL, J. VAYSSIÈRE. *A security framework for reflective Java applications*, in "Software: Practice and Experience", 2003, vol. 33, n° 9, pp. 821–846, http://dx.doi.org/10.1002/spe.528

[55] P. COINTE. *Metaclasses are First Class: the ObjVlisp Model*, in "Proceedings OOPSLA '87, ACM SIGPLAN Notices", December 1987, vol. 22, pp. 156–167

[56] S. DENIER. *Traits Programming with AspectJ*, in "Actes de la Première Journée Francophone sur le Développement du Logiciel par Aspects (JFDLPA'04)", Paris, France, P. COINTE (editor), September 2004, pp. 62–78

[57] S. DUCASSE, T. GÎRBA. *Using Smalltalk as a Reflective Executable Meta-Language*, in "International Conference on Model Driven Engineering Languages and Systems (Models/UML 2006)", Berlin, Germany, LNCS, Springer-Verlag, 2006, vol. 4199, pp. 604–618 [*DOI :* 10.1007/11880240_42], http://scg.unibe.ch/archive/papers/Duca06dMOOSEMODELS2006.pdf

[58] S. DUCASSE, T. GÎRBA, M. LANZA, S. DEMEYER. *Moose: a Collaborative and Extensible Reengineering Environment*, in "Tools for Software Maintenance and Reengineering", Milano, RCOST / Software Technology Series, Franco Angeli, 2005, pp. 55–71, http://scg.unibe.ch/archive/papers/Duca05aMooseBookChapter.pdf

[59] S. DUCASSE, O. NIERSTRASZ, N. SCHÄRLI, R. WUYTS, A. P. BLACK. *Traits: A Mechanism for fine-grained Reuse*, in "ACM Transactions on Programming Languages and Systems (TOPLAS)", March 2006, vol. 28, n° 2, pp. 331–388 [*DOI :* 10.1145/1119479.1119483], http://scg.unibe.ch/archive/papers/Duca06bTOPLASTraits.pdf

[60] S. DUCASSE, R. WUYTS, A. BERGEL, O. NIERSTRASZ. *User-Changeable Visibility: Resolving Unanticipated Name Clashes in Traits*, in "Proceedings of 22nd International Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA'07)", New York, NY, USA, ACM Press, October 2007, pp. 171–190 [*DOI :* 10.1145/1297027.1297040], http://scg.unibe.ch/archive/papers/Duca07b-FreezableTrait.pdf

[61] A. DUNSMORE, M. ROPER, M. WOOD. *Object-Oriented Inspection in the Face of Delocalisation*, in "Proceedings of ICSE '00 (22nd International Conference on Software Engineering)", ACM Press, 2000, pp. 467–476

[62] K. FISHER, J. REPPY. *Statically typed traits*, University of Chicago, Department of Computer Science, December 2003, n° TR-2003-13, http://www.cs.uchicago.edu/research/publications/techreports/TR-2003-13

[63] P. W. L. FONG, C. ZHANG. *Capabilities as alias control: Secure cooperation in dynamically extensible systems*, Department of Computer Science, University of Regina, 2004

[64] M. FURR, J.-H. AN, J. S. FOSTER. *Profile-guided static typing for dynamic scripting languages*, in "OOPSLA'09", 2009

[65] A. GOLDBERG. *Smalltalk 80: the Interactive Programming Environment*, Addison Wesley, Reading, Mass., 1984

[66] L. GONG. *New security architectural directions for Java*, in "compcon", 1997, vol. 0, 97 p. , http://dx.doi.org/10.1109/CMPCON.1997.584679

[67] M. HICKS, S. NETTLES. *Dynamic software updating*, in "ACM Transactions on Programming Languages and Systems", nov 2005, vol. 27, n° 6, pp. 1049–1096, http://dx.doi.org/10.1145/1108970.1108971

[68] G. KICZALES, J. DES RIVIÈRES, D. G. BOBROW. *The Art of the Metaobject Protocol*, MIT Press, 1991

[69] G. KICZALES, L. RODRIGUEZ. *Efficient Method Dispatch in PCL*, in "Proceedings of ACM conference on Lisp and Functional Programming", Nice, 1990, pp. 99–105

[70] R. KOSCHKE. *Atomic Architectural Component Recovery for Program Understanding and Evolution*, Universität Stuttgart, 2000, http://www2.informatik.uni-stuttgart.de/cgi-bin/NCSTRL/NCSTRL_view.pl?id=DIS-2000-05&mod=0&engl=0&inst=PS

[71] S. LIANG, G. BRACHA. *Dynamic Class Loading in the Java Virtual Machine*, in "Proceedings of OOPSLA '98, ACM SIGPLAN Notices", 1998, pp. 36–44

[72] L. LIQUORI, A. SPIWACK. *FeatherTrait: A Modest Extension of Featherweight Java*, in "ACM Transactions on Programming Languages and Systems (TOPLAS)", 2008, vol. 30, n° 2, pp. 1–32 [*DOI :* 10.1145/1330017.1330022], http://www-sop.inria.fr/members/Luigi.Liquori/PAPERS/toplas-07.pdf

[73] B. LIVSHITS, T. ZIMMERMANN. *DynaMine: finding common error patterns by mining software revision histories*, in "SIGSOFT Software Engineering Notes", September 2005, vol. 30, n° 5, pp. 296-305

[74] R. C. MARTIN. *Agile Software Development. Principles, Patterns, and Practices*, Prentice-Hall, 2002

[75] M. S. MILLER. *Robust Composition: Towards a Unified Approach to Access Control and Concurrency Control*, Johns Hopkins University, Baltimore, Maryland, USA, May 2006

[76] M. S. MILLER, C. MORNINGSTAR, B. FRANTZ. *Capability-based Financial Instruments*, in "FC '00: Proceedings of the 4th International Conference on Financial Cryptography", Springer-Verlag, 2001, vol. 1962, pp. 349–378

[77] O. NIERSTRASZ, S. DUCASSE, N. SCHÄRLI. *Flattening Traits*, in "Journal of Object Technology", May 2006, vol. 5, n° 4, pp. 129–148, http://www.jot.fm/issues/issue_2006_05/article4

[78] P. J. QUITSLUND. *Java Traits — Improving Opportunities for Reuse*, OGI School of Science & Engineering, Beaverton, Oregon, USA, September 2004, n° CSE-04-005

[79] J. REPPY, A. TURON. *A Foundation for Trait-based Metaprogramming*, in "International Workshop on Foundations and Developments of Object-Oriented Languages", 2006

[80] F. RIVARD. *Pour un lien d'instanciation dynamique dans les langages à classes*, in "JFLA96", Inria — collection didactique, January 1996

[81] J. H. SALTZER, M. D. SCHOROEDER. *The Protection of Information in Computer Systems*, in "Fourth ACM Symposium on Operating System Principles", IEEE, September 1975, vol. 63, pp. 1278–1308

[82] N. SANGAL, E. JORDAN, V. SINHA, D. JACKSON. *Using Dependency Models to Manage Complex Software Architecture*, in "Proceedings of OOPSLA'05", 2005, pp. 167–176

[83] N. SCHÄRLI, A. P. BLACK, S. DUCASSE. *Object-oriented Encapsulation for Dynamically Typed Languages*, in "Proceedings of 18th International Conference on Object-Oriented Programming Systems, Languages and Applications (OOPSLA'04)", October 2004, pp. 130–149 [*DOI :* 10.1145/1028976.1028988], http://scg.unibe.ch/archive/papers/Scha04bOOEncapsulation.pdf

[84] N. SCHÄRLI, S. DUCASSE, O. NIERSTRASZ, A. P. BLACK. *Traits: Composable Units of Behavior*, in "Proceedings of European Conference on Object-Oriented Programming (ECOOP'03)", LNCS, Springer Verlag, July 2003, vol. 2743, pp. 248–274 [*DOI :* 10.1007/B11832], http://scg.unibe.ch/archive/papers/Scha03aTraits.pdf

[85] C. SMITH, S. DROSSOPOULOU. *Chai: Typed Traits in Java*, in "Proceedings ECOOP 2005", 2005

[86] G. SNELTING, F. TIP. *Reengineering Class Hierarchies using Concept Analysis*, in "ACM Trans. Programming Languages and Systems", 1998

[87] K. J. SULLIVAN, W. G. GRISWOLD, Y. CAI, B. HALLEN. *The Structure and Value of Modularity in SOftware Design*, in "ESEC/FSE 2001", 2001

[88] D. VAINSENCHER. *MudPie: layers in the ball of mud*, in "Computer Languages, Systems & Structures", 2004, vol. 30, n$^{o}$ 1-2, pp. 5–19

[89] N. WILDE, R. HUITT. *Maintenance Support for Object-Oriented Programs*, in "IEEE Transactions on Software Engineering", December 1992, vol. SE-18, n$^{o}$ 12, pp. 1038–1044